US Army Corps
of Engineers
Construction Engineering
Research Laboratory

USACERL Technical Report P-91/49
August 1991

2

The Engineer Module of the Vector-In-Commander (VIC) Battle Simulation

by
Carol A. Subick

This report documents the enhancements made by the U.S. Army Construction Engineering Research Laboratory (USACERL) to the Vector-In-Commander (VIC) battle simulation under the Engineer Model Improvement Program (EMIP). The engineer module of VIC version 2.0 has been completely redesigned and coded. This report describes the methodology, data structures, input requirements, and output specification, summarizes routines and events, and presents a calling tree for the new engineer module.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 1991	Final

4. TITLE AND SUBTITLE
The Engineer Module of the Vector-In-Commander (VIC) Battle Simulation

5. FUNDING NUMBERS
PE 4A162734
PR AT41
WU SE-Y21

6. AUTHOR(S)
Carol A. Subick

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Construction Engineering Research Laboratory (USACERL)
2902 Newmark Drive, PO Box 9005
Champaign, IL 61826-9005

8. PERFORMING ORGANIZATION REPORT NUMBER
TR P-91/49

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Chief of Engineers
ATTN: DAEN-ZA
20 Massachusetts Avenue, NW.
Washington, DC 20314-1000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
Copies are available from the National Technical Information Service, 5285 Port Royal Road, Springfield, VA 22161.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This report documents the enhancements made by the U.S, Army Construction Engineering Research Laboratory (USACERL) to the Vector-In-Commander (VIC) battle simulation under the Engineer Model Improvement Program (EMIP). The engineer module of VIC version 2.0 has been completely redesigned and coded. This report describes the methodology, data structures, input requirements, and output specification, summarizes routines and events, and presents a calling tree for the new engineer module.

14. SUBJECT TERMS
Vector-In-Commander
Engineer Model Improvement Program

15. NUMBER OF PAGES
146

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	SAR

FOREWORD

This work was performed for the Office of the Chief of Engineers (OCE) under Project 4A162734AT41, "Military Facilities Engineering Technology"; Work Unit SE-Y21, "VIC-Based Engineer FAM (AMIP)." The technical monitor was MAJ Dave Davis, ATSE-CDC-M, U.S. Army Engineer School.

The research was conducted by the Facility Systems Division, U.S. Army Construction Engineering Research Laboratory (USACERL-FS). Dr. Michael J. O'Connor is Chief of USACERL-FS. The USACERL technical editor was Linda L. Wheatley, Information Management Office.

COL Everett R. Thomas is Commander and Director of USACERL, and Dr. L.R. Shaffer is Technical Director.

CONTENTS

CONTENTS (Cont'd)

THE ENGINEER MODULE OF THE VECTOR-IN-COMMANDER (VIC) BATTLE SIMULATION

1 INTRODUCTION

Background

The Engineer Model Improvement Program (EMIP) was established in 1988 as a comprehensive effort to ensure that engineers are properly represented in the Army's land combat models. The focus of EMIP's first phase was the hierarchy of analytical models of the Army Model Improvement Program (AMIP): the Combined Arms and Support Task Force Evaluation Model (CASTFOREM), the Vector-In-Commander (VIC) battle simulation, and the Force Evaluation Model (FORCEM). The EMIP plan, published by the Engineer Studies Center in August 1988,[1] outlined improvements to the engineer representation in each of the three models. The plan identified the VIC enhancements as the most important of the three efforts because VIC's level of resolution encompasses the most complete battlefield environment for studying combat engineer mission areas. The EMIP plan proposed that the resulting model would be capable of serving as the first accredited engineer functional area model (EFAM) as well as providing the Army with a much improved analytic tool.

Approach

In Fiscal Year (FY) 89, the U.S. Army Construction Engineering Research Laboratory (USACERL) began implementing a new representation of the combat engineer function in VIC. At the same time, the U.S. Army Engineer Waterways Experiment Station (USAEWES) launched an effort to improve VIC's representations of unit movement and the effects of terrain and obstacles. Teams at the two Corps of Engineers laboratories coordinated their efforts to ensure a consistent, logical development stream. The focus of the USACERL team was the design and implementation of a completely new engineer module for VIC. This new module was integrated with the changes proposed by USAEWES and with the other VIC modules. The final product is computer source code for the VIC engineer module with documentation as required by the VIC Configuration Management Charter. The computer code was tested in a model verification and validation study by an independent team at USAEWES.

This report describes VIC's new engineer module, including its basic design features, data structures and procedures, input data requirements, and output structure.

Objective

The objective of this work is to ensure that VIC represents combat engineer operations well enough to allow analysts to accurately measure both the impact of changes in force structure on engineer capability and the contribution of engineers to the combined-arms team.

---

[1] Larry Wright, *The Engineer Model Improvement Program Plan*, USAESC-R-88-6 (U.S. Army Engineer Studies Center, August 1988).

Mode of Technology Transfer

The computer source code, documentation, and required configuration control reports resulting from this effort were delivered to the VIC model proponent, U.S. Army Training and Doctrine Command (TRADOC) Analysis Command at Fort Leavenworth (TRAC-FLVN), in December 1990.

2 METHODOLOGY

Overview

VIC is a two-sided deterministic simulation of combat in a combined-arms environment designed specifically to study Airland Battle concepts in a variety of scenarios. VIC represents the major elements of land and air forces at the U.S. Army corps level with a commensurate enemy force in a mid-intensity battle. According to Army Regulation 5-11,[2] VIC is used to: design force structures and develop concepts, doctrine, and tactics for brigades, divisions, and corps; determine corps/division resource requirements for sustained combat operations; and study materiel and item systems that are organic to, or have profound influence on, the capabilities of brigades, divisions, or corps.

The variety of questions studied with VIC requires that the model contain a fairly comprehensive representation of the combat, combat support, and combat service support elements. Through input data, the scenario developer describes the force composition and maneuver plan of the opposing sides and determines the terrain upon which the forces move and fight. The assumption at this point is that the input data sets the stage for a simulated battlefield environment that provides sufficient realism and detail to test alternative courses of action. As a discrete event simulation, VIC builds this very complex battlefield environment one event at a time, using procedures in each event to make relatively simple decisions about what happens next.

The decisions made at the occurrence of a simulation event are based on the state of the elements pertinent to the event and on their interactions with one another. The model builder's task in abstracting a real-world system is to decide which elements of the system are pertinent and to represent them and their interactions in a way that allows the model to reflect the real world as closely as possible, or at least as closely as the model users require. This process is infinitely flexible, with no right or wrong way to identify the essential elements of the real world to include in the model and even less constraint on how to represent the very complex interactions of the few elements actually chosen. The general approach to building a model is to start with a simple representation and, as experience in using it grows, gradually improve its faithfulness to the modeled system through a series of refinements and enhancements. The development of VIC has followed this pattern, from its origins as an amalgam of simpler models, Vector and Commander, to the current yearly release of its reference version.

This was also the approach taken by EMIP in its plan to improve VIC's combat engineer representation. In a 1987 project funded by the Army Model Management Office (now the Model Improvement and Study Management Agency), a USACERL team analyzed the engineer representation of VIC 2.0, and their report[3] identified a number of areas in which essential elements of the combat engineer function were omitted or were portrayed in a way that did not accurately reflect their real-world counterparts. The USACERL report was the basis for a VIC enhancement plan formulated by the Engineer Studies Center and published as a part of the *Engineer Model Improvement Program Plan.*[4] The goal was to improve the realism of VIC's portrayal of the combat engineer function without adding

[2] Army Regulation (AR) 5-11, *Army Model Improvement Program* (Headquarters, Department of the Army [HQDA], 15 August 1983).

[3] Mark A. Fiddes and John W. Boyd, *Analysis of the Engineer Representation in the Vector in Commander (VIC) Battle Simulation Model*, Technical Report (TR) P-88/25 (U.S. Army Construction Engineering Research Laboratory [USACERL], August 1988).

[4] Larry Wright, August 1988.

a level of detail inconsistent with the resolution of the other modules and without seriously affecting run time or memory requirements.

The EMIP plan identified design improvements and new capabilities to be added to VIC during the course of a 2-year development effort. The improvements fell into three logical areas:

1. A more complete representation of the types of tasks engineers perform, including adding task types not represented in VIC 2.0 and improving the manner in which individual tasks are generated

2. A new representation of engineer units, resources, and processes to allow a more accurate assessment of engineer capabilities

3. A more detailed representation of the terrain features altered by engineers and an improved representation of maneuver unit interactions with those features.

In VIC's modular design, the entities and processes of the first two areas above fall almost entirely within the domain of the engineer (EN) module, while those of the third area are almost independent of the EN module. With this in mind, the EMIP plan divided the responsibility for the VIC improvements between two Corps of Engineers laboratories: USACERL and USAEWES. USACERL's responsibility was to improve the EN module's representation of combat engineers, i.e., the representation of engineer units, resources, methods, missions, and task performance. USAEWES's responsibility was to improve VIC's representation of unit movement, including the use of roads and the effects of terrain and engineer obstacles.

The USACERL team quickly concluded that the VIC 2.0 EN module design structures could not be adapted to accommodate the proposed changes. Design and implementation of a completely new engineer module began in FY89. By the end of the year, the basic structures for representing engineer units, resources, and task performance were complete, and an interim version of the new EN module was included in the release of VIC 3.0. During the second year of development, researchers concentrated on improving several key engineer processes (e.g., mission assignment and planning), increasing efficiency and flexibility, and adding new task type capabilities. The resulting version of the EN module has been integrated into the 1991 release of VIC 4.0. Our objective here is to describe the basic design elements of the new engineer representation in VIC 4.0.

The Combat Engineer Function

Understanding the structure of the EN module requires a familiarity with the real-world combat engineers it attempts to model. The role of the combat engineer is described succinctly in FM 5-100, *Engineer Combat Operations*:[5]

> Engineers adapt terrain to multiply the battle effectiveness of fire and maneuver. This engineer component of the close combat triad (fire, maneuver, terrain) is described within the five engineer functional areas: mobility, countermobility, survivability, sustainment engineering, and topographic engineering. Mobility frees the commander from movement limitations imposed by natural terrain or enemy action to allow maneuver of tactical units into positions of advantage. Countermobility directly attacks the enemy commander's ability to execute his plan where and when he desires. Survivability allows friendly forces to fight from locations that would otherwise be untenable. Sustainment engineering adds depth in space and time to the battle by ensuring that sustainment operations can

[5] *Engineer Combat Operations*, Field Manual (FM) 5-100 (HQDA, 1988).

occur. Topographic engineering defines and delineates the terrain for planning and operations, and provides precise location data to modern efficient weapons systems. To accomplish these functions, engineers serve throughout the theater, though the bulk of engineer forces are forward within the close battle area. As with all arms, engineers are integrated into the scheme of maneuver and are massed at points critical to the battle.

The mobility function includes all efforts required to allow the fighting force to move at will. Engineer terrain analysis and reconnaissance identifies the best routes for movement, and engineers assigned to lead elements provide rapid, in-stride breaching of obstacles. Obstacles may be natural (e.g., rivers), cultural (e.g., embankments), or reinforcing (e.g., minefields and antitank ditches [ATDs]). The mobility function also includes construction of combat trails through areas where routes do not exist and forward aviation combat engineering (FACE), involving the expedient development and repair of landing strips, low altitude parachute extraction systems, and forward arming and refueling points (FARPs).

The countermobility function includes all efforts aimed at restricting enemy movement. Engineers emplace obstacles to reduce the enemy's ability to maneuver, to increase the enemy's vulnerability to direct and indirect fire, and to protect friendly forces from counterattack. Tactical obstacles include minefields, destroyed bridges, ATDs, wire entanglements, abatis, etc. Such obstacles may be used individually or as components of an integrated obstacle system.

The engineer survivability function includes all efforts at protecting personnel, weapons, and supplies from exposure to both direct and indirect fire. The effort focuses primarily on the construction of protective positions for combat vehicles, direct fire weapons, artillery and air defense systems, command and control elements, and critical logistics assets. Also included is the proper use of camouflage and deception to conceal the location of key force components.

The engineer sustainment function includes all efforts required to sustain the fighting force. This includes replacing assault and tactical bridging with fixed bridging, clearing minefields and removing other obstacles, maintaining and improving lines of communication, constructing and repairing airfields and aircraft facilities, and constructing and repairing support facilities.

The topographic engineering function includes terrain analysis, production of updated maps and overlays, and survey support for artillery and missile targeting requirements.

While this description applies mostly to U.S. Army engineer mission areas, the combat engineer elements of allied and threat forces are similarly tasked.

The Essential Elements of the Combat Engineer Model

The starting point for building a combat engineer model was this list of engineer battlefield tasks:

Mobility

- Breach minefields
- Breach line obstacles
- Breach obstacle complexes
- Construct combat trails
- Terrain analysis/reconnaissance
- Tasks related to FACE

Countermobility

- Emplace minefields
- Emplace line obstacles
- Emplace obstacle complexes
- Destroy line breaches

Survivability

- Prepare protective positions
- Employ camouflage and deception

Sustainment (General Engineering)

- Clear minefields
- Improve line obstacle breaches
- Maintain/repair roads
- Construct/repair air facilities
- Construct/repair CSS facilities

Topographic Engineering

- Terrain analysis
- Map production
- Survey support

Action is what simulation is all about, and the engineer tasks form the nucleus of combat engineer actions. Each action has two relatively independent components: how the task is done and what happens when the task is complete. The primary goal was to improve VIC's realism without adding too much detail. The USACERL researchers made several decisions that became basic criteria for choosing the task types to model and for designing the core structures of VIC's engineer representation.

Decision 1: Modeling the effect of engineer task performance is more important than modeling the task performance itself. For example, being able to represent minefields on the battlefield and unit interactions with them is more important than representing how the minefields were put there in the first place.

Decision 2: If the effect of an engineer task cannot be modeled, then the performance of the task should not be modeled. This is because the level and priority of engineer effort are generally functions of the effect and, therefore, cannot be calculated realistically in the absence of an effect. For example, if the condition of an air unit's runways does not affect the unit's ability to accomplish its mission, then the level of effort and the priority for keeping them in operation cannot be based on the simulated situation.

Decision 3: If the simulation contains no natural means for recognizing the need for an engineer task, then explicitly modeling the task performance has no value. For example, if roads cannot be damaged by friendly or enemy action, then repair of roads should not be modeled explicitly.

Actions require objects to receive them, so the next logical step was to consider the representation of the physical features that are altered during the performance of engineer tasks:

- man-made obstacles like minefields and ATDs
- natural/cultural obstacles like rivers and embankments
- fortified positions
- combat trails
- bridges
- roads
- air facilities like runways and hangars
- CSS facilities like warehouses and supply depots
- terrain.

These physical features serve as the link between the engineer effort and the engineer effect. The model's representation of a physical feature is the key to the level of detail possible in representing both the effort involved in accomplishing the associated task and the effect of its completion.

Combining the original decision criteria with the physical feature structures that VIC gave the researchers to build on, the engineer task types were separated into two lists according to whether or not both the need for such a task and the effect of such a task could be modeled in VIC:

	Included	**Excluded**
Mobility		
	Breach minefields	Terrain analysis/reconnaissance
	Breach line obstacles	Tasks related to FACE
	Breach obstacle complexes	
	Construct combat trails	
Countermobility		
	Emplace minefields	Crater roads
	Emplace line obstacles	
	Emplace obstacle complexes	
	Destroy line obstacle breaches	
Survivability		
	Prepare protective positions	Employ camouflage and deception
Sustainment (General Engineering)		
	Clear minefields	Construct/repair air facilities
	Improve line breaches	Construct/repair CSS facilities
	Maintain roads	Repair road craters
Topographic Engineering		
		Terrain analysis
		Map production
		Survey support

Modeling the effect of engineer activity is extremely task-specific. Ground units must be able to recognize the presence of the physical feature altered by the activity and to react to it appropriately. Each unit must suffer delay and attrition from encountered obstacles according to that unit's ability to breach or bypass; and each type of obstacle (minefield, line obstacle, complex) must have its own distinct algorithms for computing encounters, tactics, delays, attrition, and future effectiveness. Units must use prepared positions properly and have appropriate attrition in a direct fire attack. Combat trails must speed unit movement through difficult terrain. Obstacle breaches must decrease the delay and attrition of an obstacle. And the condition of a road must affect the speed of units using it. Once the individual algorithms for computing the delays, attrition, speed, etc. are developed and accepted, the most important element for modeling the effect of engineer activity is correctly controlling it through input data.

Enhancements to VIC to allow a realistic representation of the effect of engineer activity included developing new structures for the physical features altered by the activity and new algorithms for determining what happens to units when they encounter such features. The USACERL effort included the complete development of line obstacle breaches, combat trails, and prepared positions. Responsibilities for the development of minefields, obstacle complexes, and roads had been divided between USAEWES and USACERL, so development teams from the two labs formed one work team to design the required data structures and algorithms for those features. The next section of this chapter presents a detailed description of each implementation.

While modeling the effect of engineer task performance is crucial to the realism of the simulated battlefield environment, modeling the engineer processes related to that task performance is not. Early in the EMIP effort, concerns were raised that the proposed restructuring of the EN module would introduce an inconsistent level of detail to VIC. This concern warrants a close look at how the engineer processes were ultimately represented in the new EN module and why that representation is consistent with VIC's model of corps-level combined arms combat.

Overcoming obstacles, digging in to hold a position, controlling enemy movement through the use of obstacles, and keeping the lines of communication open are all vital in determining how ground forces move and fight. Whether a combat force can accomplish each of those tasks—more simply, the timing of the task accomplishment—greatly influences the flow of battle. VIC can be configured to assume that each requested engineer task will always be accomplished at the requested time. The goal was to extend VIC to allow the scenario data and the situation that evolves from it to control if and when each task is accomplished; that is, to make the ability to accomplish engineer tasks sensitive to the input descriptions of the factors which logically affect that ability: force structure, resources, locations of relevant elements, battlefield terrain, the environmental and tactical situations, etc.

Decision 4: The primary reason for explicitly modeling engineer task performance is to improve the accuracy of our estimates of engineer capability, i.e., what engineer tasks can be done and when. The assumption was that, with VIC's resolution of maneuver units, the battlefield impact of engineers is in the timing and the effect of their task completion and not in their presence *per se* or in the simulated performance of their tasks.

Ultimately, most of the processes of the EN module are devoted to determining a realistic answer to the question: When will the requested task be completed? Any number of factors may be considered in determining the answer, according to what is taken into account. For example, the time required to move the equipment to the work site might be considered. An estimate of that time could be based simply on the speed of the equipment and the distance it needs to travel. A different estimate might include considerations for the intervening terrain, obstacles to be encountered, delays due to enemy attack, etc.

The accuracy of an estimate is a function of the number of influencing factors one takes into account and the accuracy of the estimates of their individual impacts.

The EN module of VIC 2.0 linked the timing of engineer task completion to the scenario and its evolving situation, but its basic design failed to take into account some of the most crucial situational factors. The main obstacle to enhancing VIC's original EN module was the inflexibility of its design structures, especially the fact that its four task types (emplace minefield, emplace line obstacle, prepare position, and breach line obstacle) were separately represented, each with its own job list, performing work teams, and performance method. Since the researchers wanted to add new task types, they could not use the original EN module without adding unnecessarily complicated structures. A crucial decision was made to simplify the engineer processes in the design by modeling the performance of all of the task types in a single, generic way.

Decision 5: The basic processes in representing engineer task performance are the same for all task types and can be modeled by a single sequence of procedures.

Obviously, each type of task has its own mix of required equipment and supplies, work durations, and dependencies on environmental factors (e.g., weather and visibility). But the actions taken to accomplish a task are essentially the same for all types of tasks:

1. Recognize the need to perform the task
2. Assign the task to a unit and determine how the task is to be done
3. Check that the required equipment and supplies are available
4. Determine how long it will take to move the equipment and supplies to the site and perform the task
5. Expend the required supplies and make the equipment unavailable for use on another task until the current task is complete
6. Register the effect of work completion at the end of the calculated work time.

These six steps helped to isolate the impact of the crucial factors that were to be taken into account. Assigning the task to a unit was linked to the scenario's command structures and resource allocations. Determining how the task was to be done was linked to a new structure, the engineer technique, which allowed input data to describe more than one way to do a task and to link equipment requirements to the method for doing the task rather than to the task itself. Checking for the availability of the required equipment and supplies was linked to the equipment/supply list of the chosen technique, the assigned unit's job load and supply inventories, and the task's relative priority. Determining how much time is required to complete the task was linked to the timing factors of the chosen technique, the terrain, the environmental and tactical situations, and equipment levels.

Breaking the problem down to this six-step sequence gave tremendous flexibility in controlling the level of detail at which engineer processes are modeled. The design of the new engineer technique structure, which linked engineer resources with task performance requirements, provided a mechanism for modeling the engineer resources at any desired level of detail. More importantly, the engineer technique was the fundamental structure upon which we based two distinct methods for representing steps 2 through 5 of the engineer processing sequence. One method uses a detailed simulation of the process to determine the task completion time, while the other uses a "best case" estimate of the completion time with no simulation of the process. The two methods came to be known as the explicit and the implicit representation of engineers.

The implicit representation models an implied engineer presence or a nonengineer capability. In the implicit representation, a nonengineer unit owns the resources and "performs" its own task, with the

estimate of the completion time based on the current state of the crucial factors (available equipment, weather, day/night, enemy location) and on a travel time dependent only on equipment speed and distance. In the explicit representation, an engineer unit owns the resources and performs the task for a supported unit. The estimate of the explicit task completion time is made by actually simulating the task performance process: forming a work team, moving it across the terrain to the site, and adjusting work times as the situation changes.

The implicit engineer representation grew out of an early EMIP requirement:

> **Decision 6: The ability to perform engineer tasks is not restricted to engineer units, so task performance by nonengineer units must be allowed. For example, artillery can emplace minefields, and tanks with plows or rollers can breach them; infantry can dig themselves in; air missions can target bridges for demolition.**

While the engineer technique structure was originally designed to allow the engineer processes to be simulated in detail, that same structure provided the essential link between a unit's resources and its "engineer" capabilities. Determining if and when any given unit can accomplish a task reduces to comparing its resources with the required equipment lists of appropriate techniques and computing the work time from the technique data if a usable technique is found. If the given unit is an engineer, the task performance process is simulated completely. If the given unit is not an engineer, the completion time is estimated using current conditions, and the task completion is scheduled for the appropriate time without any simulation.

With the engineer technique data in hand, the scenario developer can enable a unit to perform a certain type of engineer task simply by giving that unit the required equipment (weapons) and supplies for an appropriate technique. At the same time, the level of detail that the model will use for simulating task performance is determined by the type of unit receiving the engineer resources, implicit for nonengineer and explicit for engineer. To complete the flexibility of the engineer module, both implicit and explicit representations may be used in the same scenario.

The new engineer module is designed to allow VIC users to see as much or as little of the combat engineer function as they wish. To use the new module properly and to take full advantage of its flexibility, the user must understand the implementation of each of these essential design elements: the representation of the physical features altered by engineers, the effects of engineer action on the associated features, and the engineer task performance sequence as it is represented implicitly and explicitly.

The Physical Features Altered by Engineers

Minefields

VIC already contained an entity to represent an individual minefield and an entity to represent a minefield type. Each minefield has a frontage, depth, orientation, number and type of mines, portion cleared, and attributes to track how it was emplaced, who knows about it, and whether or not it is active.

Individual minefields are specified in the minefield (MF) input data as a part of each side's obstacle plan. The input data indicate whether the minefield is to be pre-emplaced (i.e., emplaced without engineer effort) or hand-emplaced (i.e., emplaced with engineer effort). Input data also specifies a time for the emplacement, which is the activation time for pre-emplaced minefields and the latest completion time for hand-emplaced minefields. If a minefield is to be hand-emplaced, an early start time for the engineer effort is specified, and the option is available to associate the minefield with a given ground unit. The linking of a ground unit with a minefield controls the assignment of the emplacement task, allowing the

specified ground unit to accomplish the task implicitly if it is capable and considering engineer units in support of the given unit to accomplish the task explicitly.

The effect of the engineer task to emplace a minefield is simply to activate it. A moving unit considers all minefields in it path but stops only at active minefields emplaced either by the enemy or by air/artillery. If an encountering unit's decision is to breach the minefield, an algorithm computes how long it will take that unit to do so without engineer assistance. If the computed time can be bettered by the encountering unit using implicit engineer play or by that unit's engineer support using explicit engineer play, then the task is done by the appropriate method. The benefit to the encountering unit is a shortened breach delay, while following units find a breached minefield.

The EMIP plan proposed the addition of an ability to dynamically generate the need for new minefields and other obstacles in response to the evolving situation. We decided that VIC's terrain resolution was not sufficient to automate the required decision process and that controlling the placement of new minefields by adding the early start time to the minefield data accomplished the desired engineer responsiveness in VIC. This also gave control of the obstacle plan to the scenario developer, who has access to the type of detailed information required to place the obstacles properly.

Line Obstacles

VIC already contained entities to represent individual line obstacles, entities to represent single segments of a line obstacle, and entities to represent line obstacle types. The combined attributes of these entities provided only enough information to know the obstacle location, the breached and unbreached delays associated with crossing one of its segments, whether or not a segment is breached, and whether breaching a segment requires engineers. VIC did not have an entity to represent a breach for a line obstacle but represented a breached segment by changing one of the segment's attributes.

Because of the importance of river crossing operations to the engineer mission, the representation of a line obstacle breach was changed by adding a new entity. The obstacle breach point has a specific location on a segment and an effectiveness factor used in determining the associated crossing delay. The implementation of this new breach point entity changes much of the way in which line obstacle encounters are represented. A single segment may now have more than one breach, and an encountering unit may search for and use all breach points within its deployment (or within 5 kilometers) along the encountered segment and adjacent segments. See *Bridges* (p 16) for details about implementation of obstacle breach points.

Line obstacle types and individual line obstacles are specified in the terrain-barrier (TB) data. Line obstacles are natural, cultural, or man-made. Each type of line obstacle has associated breached and unbreached delays for each side. Data for the man-made obstacles indicate the emplacing side and whether or not engineer effort is required. As with the minefield plan, engineer emplacements have a specified early start time and can be associated with a ground unit. The data for each line obstacle also indicates the location of preexisting breaches/bridges and a possible plan for their destruction (see *Bridges*, p 16).

The effect of the engineer task to emplace a line obstacle is to make it active. A moving unit considers all line obstacles in its path but stops only at active obstacles not owned by its side. An encountering unit searches along the obstacle for available breach points and calculates its delay according to what it finds. If no breach point exists, the unit considers its own engineer capabilities to breach implicitly and, failing that, generates a breaching mission for explicit engineer performance. The end result of a line obstacle breaching mission varies according to the technique used. In all cases, a search is made for all units stopped at the obstacle to allow the new breach point to shorten their delay. If the breaching technique used does not require retrievable assets, then a breach point is created. When

retrievable assets are used in an explicit engineer breach, then a breach point is created, and a mission is generated to improve the breach and retrieve the assets. When retrievable assets are used in an implicit breach, no breach point is created.

Bridges

The new breach point entity provides the structure needed to greatly improve VIC's representation of river crossing operations and bridge destruction. The breach point entity has a specific location on a line obstacle segment and a relative effectiveness which helps determine its crossing delay.

Breach points are created in one of three ways: in the TB data when line obstacles are specified, in the logistics (LO) data when the road network is linked to the terrain by creating breach points at each intersection of a road and a line obstacle, and as the result of an effort to breach a line obstacle.

Breach points are improved by engineers in two situations: the breach point was created as a result of an engineer mission employing retrievable assets and the breach point is part of the road network and has been destroyed using the external event EN.BRIDGE.DEMOLITION. Improving a breach changes the relative effectiveness of the breach point.

Destruction of breach points is controlled either through the TB input data at the time the breach point is created or through the external event EN.BRIDGE.DEMOLITION. This event can be used to destroy bridges along an avenue of approach or on a road network or to retrieve assault bridging for use at another location. The user may specify that the bridge be destroyed at a particular time or by the crossing of a particular unit and may specify that the task be accomplished with or without engineer effort.

The format for the external event is a line of seven fields in the scenario's external file:

Field 1 - "EN.BRIDGE.DEMOLITION" to identify the event

Field 2 - DD HH MM to specify the time for the event in day-hour-minute format

Field 3 - X,Y or UTM coordinates of the center location of the area where bridges are to be destroyed

Field 4 - Range, in kilometers, as the radius of a circle with center given in Field 3 within which all bridges are included in this event

Field 5 - Indicator Flag to specify the action to be taken with each bridge found within range of the given location:

X - remove the bridge at the specified time without engineer effort

PB,PR - declare the bridge prepared for demolition without engineer effort for the side indicated by the second character

EB,ER - request engineers to prepare the bridge for demolition for the side indicated by the second character

DB,DR - if the bridge has been prepared by engineers for the side indicated by the second letter, then remove the bridge now; otherwise, request engineers to prepare and remove the bridge as soon as possible

U - mark the bridge so that the crossing of a certain unit will cause the bridge to be destroyed if it has been prepared

Field 6 - Unit name given only if Field 5 was a "U"

Field 7 - "*" to mark end of external event.

When a breach point is created in the TB input data by placing a "B" between the coordinates of the end points of the segment containing it, the two letter symbols used in field 5 above may be substituted for the "B" to achieve a similar effect for bridges. Our assumption regarding the preparation and demolition of a bridge is that all of the effort is concentrated in the preparation while the demolition itself is simply a recognition that permission is granted to destroy it.

Obstacle Complexes

In an effort to achieve realistic obstacle play, USAEWES created a new type of obstacle, the obstacle complex, represented as a polygonal region containing a mixture of minefields and line obstacles. The individual minefields and obstacles in the complex determine the delays and attrition of the complex as a whole.

The obstacle complex types and the individual obstacle complexes are specified in the MF data module. The obstacle complex type indicates the number and type of minefields and line obstacles composing the complex and specifies by mission number whether complexes of that type should be emplaced and breached as single engineer missions or as separate minefield and line obstacle missions. The individual obstacle complexes may be pre-emplaced or emplaced by engineers. As with minefields, if the complex is to be emplaced by engineers, the input data specifies an early start time and may associate the complex with a ground unit in order to control engineer mission assignment.

An obstacle complex may be emplaced as a single mission or as a set of missions to emplace its component obstacles. If an obstacle complex is emplaced as a single mission, then the effect of mission completion is to make the complex active and all component obstacles effective. If an obstacle complex is emplaced in parts as minefield or line obstacle missions, then the effect of a mission completion is to make the complex active with the particular component obstacle effective. A moving unit encounters an active, enemy obstacle complex at the first point along its path where the unit's circle of deployment intersects the polygon of the complex. The area of the complex crossed by the unit as it moves on its path is calculated, and the ratio of that area to the total complex area determines the portion of the obstacle seen by the unit. Delays and attrition from the effective component obstacles in the complex are accumulated and scaled by the portion seen. The delay from the complex is spread along the unit's path through the complex as a degradation in the unit's speed. Discovery attrition is assessed when the unit enters the complex, and crossing attrition is assessed when the unit exits the complex.

Engineers affect the ability of a unit to move through an obstacle complex in two ways. If the unit knows about the complex before its encounter, it may request an engineer breach of the complex. Such a mission may be done as an obstacle complex breach or as a set of missions to breach the component obstacles. The decision as to which method is used is based on the prototype mission numbers for the complex and the engineer technique data. The effect of completing an obstacle complex breach is to breach every component obstacle. When the unit encounters the complex, its assessment of breaching capabilities is based on currently available equipment, and breaching activity addresses only the individual component obstacles. In crossing an obstacle complex, a unit attempts to breach the number of component

obstacles seen. For example, if the unit's path intersects half of the complex, the unit sees and needs to breach only half of each of the types of component obstacles in the complex.

USACERL's current experience with using obstacle complexes indicates that the frontage of the polygonal region should be no larger than the diameter of a typical maneuver unit and the depth should be no larger than the radius of a typical unit. Violating this rule of thumb usually leads to exaggerated delays and attrition if more than one unit is in the complex at any given time.

Prepared Positions

VIC's former representation of a prepared position as an attribute of a unit path point limited the ability to make distinctions in the levels of effort required to dig in different kinds of units. In addition, because the position was not linked to the terrain, the use of the position by more than one unit and the destruction of the position were poorly played. A new prepared position entity was created and filed as a terrain feature that has a type linked to the type and level of unit for which it is constructed.

The engineer missions to prepare positions are created when the ground unit path points are read in the global ground (GG) data. A path point may be declared as already prepared or as needing preparation by engineers. In either case, a position is created and filed in the appropriate grid square. If engineers are requested to prepare a position, a mission is generated to do so. The engineer data provides the data needed to link each unit prototype with a position prototype. The type of position to be prepared is determined by the type of unit requesting it, and the level of effort to prepare the position is determined by the engineer technique data for that specific position prototype.

A position may be prepared in one of three ways. The GG data specifies a unit prototype attribute indicating the time required by that prototype to prepare its own position. In addition, the engineer data may have techniques for preparing a particular type of position, and the technique may be used explicitly or implicitly. With the GG data, a unit's ability to prepare its own position is independent of equipment availability, and work on the position progresses only as long as the unit is at the location. With implicit preparation, a unit's ability to prepare its position depends on the availability of owned equipment, and work on the position can begin as soon as the unit passes a path point within its radius of control of the location. With explicit engineer preparation, the position may be prepared well in advance of the unit's arrival using equipment not necessarily in direct support of the unit.

Combat Trails

VIC had no representation of a combat trail but already tracked ground cover and its effect on trafficability. USACERL created a new combat trail entity, filing it with the terrain grid containing it. A simple representation of the construction task was implemented, linking engineer effort with the grid's ground cover and the dimensions of a grid cell, and linking its effect with the grid's trafficability.

Combat trails are created in the TB data and may be pre-emplaced or engineer emplaced. The location of the trail determines the grid cell containing it, and the presence of a trail is a factor in determining the trafficability of the grid. The engineer technique data for combat trail construction should specify equipment/supply requirements and work durations for constructing a trail of length equal to the length of a grid side and through the appropriate ground cover. The effect of completing a task of constructing a combat trail is to adjust the trafficability of the grid so that all units moving through the grid are aided by its presence. We assumed here that the direction of the trail through the grid was not important because major unit movements would tend to be in the same approximate orientation.

Roads

The logistics module of VIC contained a road network for use by supply convoys in moving supplies from the rear to forward supply areas. USAEWES expanded road play to allow maneuver units to use them, to tie the roads to the underlying terrain, and to track degradation in capacity as a function of use. This opened the way for modeling road maintenance, bridge destruction, and bridge improvement.

Roads and road types are specified in the LO data. Each arc of the road network has a capacity which can be degraded by use or by destruction of a bridge on the arc. When the road capacity falls below a certain use threshold, an engineer mission to maintain the road is generated. If the road capacity is affected by a destroyed bridge, cyclic processing in the LO module detects the problem and generates a mission to improve the bridge. Unlike other engineer task types, road maintenance and main supply route (MSR) bridge improvement have no specific requesting unit, so the default requesting unit is taken to be the supreme unit on the side owning the road arc. This makes implicit engineer accomplishment of the tasks difficult because the supreme unit does not usually move or have a large radius of control. USACERL's experience with road maintenance and bridge improvement suggests that both are better represented explicitly by creating a capable engineer unit to support the supreme unit.

Road repair from battle damage was omitted because VIC currently does not have any way to damage a road. If the USAEWES road enhancements lead to enemy use of friendly roads, engineers could expand their tasks to include road cratering and road repair, both easy additions to the current version of the EN module. In addition, enhancements to battlefield air interdiction (BAI) could make road damage possible by air targeting. This had been a proposed addition by White Sands Missile Range, but funding cuts canceled the project. Road damage could be controlled through an external event similar to bridge destruction; USACERL did not have sufficient resources to implement that event.

Facilities

VIC had no representation of air or CSS facilities. Unlike the other physical features altered by engineers, facilities offered no simple fix that did not violate the first three decisions above. The air and CSS functions had no dependence on facilities, so adding a representation of their facilities would have required the addition of unwarranted amounts of new data and new processing. It was decided not to explicitly model tasks related to air and CSS facilities.

Engineer Task Performance

The new EN module of VIC provides considerable flexibility in its representation of the performance of the following engineer tasks:

- breach a minefield
- breach a line obstacle
- breach an obstacle complex
- construct a combat trail
- emplace a minefield
- emplace a line obstacle
- emplace an obstacle complex
- destroy a line obstacle breach
- prepare a protective position

- clear a minefield
- improve a line breach
- maintain a road.

The scenario developer may choose to have no engineer task performance at all in a VIC run, even though the full effect of most of the task types may still be seen and measured. Input data may specify when minefields, line obstacles, obstacle complexes, and combat trails are to appear, which unit positions are already prepared, when line breaches and minefields are to disappear, and how long a unit requires to breach a minefield, line obstacle, or obstacle complex—all without engineer activity. In addition, roads can be created that require no maintenance. The task to improve a line breach is the only engineer task type that actually requires the engineer module for its representation.

Using the option to omit engineer task performance entirely requires great care and understanding on the part of the scenario developer. Since VIC's purpose is to provide a realistic battlefield environment for studying airland battle concepts, the first rule for its use should be to avoid unrealistic scenarios. In other words, units moving across VIC's battlefield should move at appropriate speeds whether or not engineers are visibly present to help them, and the terrain should include only as many man-made obstacles and positions as could reasonably be emplaced. The difficulty with using VIC without the EN module is that the scenario becomes insensitive to changes in engineer capabilities and many of the compensations in data that must be made to preserve realism are difficult to compute off-line and generally undocumented.

With the EN module in place, VIC offers a range of task performance representations. Input data controls the level of detail and the task performance representation used for each particular task. The task performance sequence is always the same:

1. Recognize the need to perform the task
2. Assign the task to a unit and determine how the task is to be done
3. Check that the required equipment and supplies are available
4. Determine how long it will take to move the equipment and supplies to the site and perform the task
5. Expend the required supplies and make the equipment unavailable for use on another task until the current task is complete
6. Register the effect of work completion at the end of the calculated work time.

The variations in the representation of this sequence occur in how each step of the sequence is accomplished. For example, step 3 is very dependent on the level of detail at which engineer equipment and engineer techniques are described in the engineer data and on what kind of unit is checking for available equipment. To understand how these variations work together, the user must have a basic understanding of the data structures for the major entities of the EN module: engineer missions, units, command structures, assets, techniques, and jobs.

Engineer Missions

As the need for engineer activity arises, whether by specific battle plan in the input data or by dynamic generation, engineer missions are created to hold the information about what needs to be done and to link together jobs to be done as a single "mission." All of the jobs which arise from one mission must be of the same type, i.e., the task and feature type must be the same for all. Only obstacle emplacement tasks may have missions with more than a single task. The significance of planning several

emplacements as one mission is that work teams will complete all tasks in the mission before returning to their headquarters unit. A system-owned variable, EN.MISSION.COUNT, is used to assign a unique sequential number to each mission, linking the jobs arising from the mission by this common number. The missions are filed in the set of missions owned by the side and represent the unassigned engineer work to be done.

Periodically, the set of missions is processed, and those missions that may be assigned to a specific engineer unit are converted to jobs and removed from the mission set. A mission can be assigned as a job if its earliest start time is before the end of the next mission processing cycle and the side has an engineer unit with the mission location in its tactical area and the capability to do the work. A mission that is impossible because the engineer input data has no technique for its task and feature type or because no engineer unit will ever be capable of doing the work is removed from the mission set and discarded. Missions which are possible but for which no engineer unit presently has the site in its tactical area or a usable technique for the given circumstances remain in the mission set. They are reprocessed at cyclic intervals and whenever an engineer unit moves to a new tactical area. The first data item in the engineer data module determines the length of the interval for processing the mission list. In order to save on run time, the interval should be in the range of 30 minutes to an hour.

Once a mission has been assigned to an engineer unit, a job is created and filed in the unit's pending list for each feature associated with the mission. The mission is removed from the side's mission list and destroyed. An engineer job is specified in terms of what needs to be done by the task type, feature type, and location (Example: "Emplace a minefield of type A at location (x,y)"). The unit responsible for the job is determined by the command structure, unit capabilities, and tactical areas, all of which are determined by input data. The priority ranking of the jobs is determined by the command structure and the situation. The possible methods for doing the job are determined by input data. The manner in which the job is actually done is determined by matching the equipment required by appropriate techniques for the given task/feature type with the equipment available to the responsible unit. The unit which does the work is a task-organized work unit created solely for the purpose of the job at hand using equipment owned by the responsible unit.

Engineer Units

The EN module has three types of engineer units to be used for the explicit task performance representation. The engineer headquarters (HQ) unit is a ground unit which is actually in the GG input data. The engineer work unit is a task-organized unit assigned to perform a specific set of jobs with equipment and supplies required for the jobs transferred to it by the HQ unit responsible for the jobs. The engineer staff unit is a HQ unit which has a nonengineer superior. The staff units are used primarily for modeling engineer command and control.

In VIC, only ground units move, fight, consume supplies, and interact. In order to model task-organized work units, extra BLUE and RED ground units of type engineer are created at the start of the simulation, marked as ..REMOVED so that they are invisible to the simulation until activated, and filed in a set of free units. Activating a free unit means removing it from the free set and using the associated ground unit for a job assignment. When the job is completed and equipment returned to its owning HQ or the work unit is destroyed, the associated ground unit is stripped of its attributes and removed; the work team is refiled in the free set for later use on other jobs. Two new data items in the GG data module specify the number of extra BLUE and RED engineer units to be created. Unlike earlier versions of VIC in which the engineer work units actually appeared in the ground unit input data, the new representation manages the organization and processing of work teams internally.

All engineer units are ground units, but only engineer HQ units have their attributes set directly by input data. As ground units which appear in the data modules, HQ units have their tactical command structure set in the ground movement (GM) data, their paths and tactical areas set in the GG data, their assets set in the front-line attrition (FL) data, and their supply needs set in the logistics (LO) data. HQ units do not do engineer work. They hold the engineer assets, are responsible for jobs as determined by simulation rules, but assign work teams to do the work with the HQ unit's assets. Each HQ unit keeps an inventory of owned equipment; one entry for each piece of equipment being tracked. For rapid processing, the HQ unit also keeps an equipment array which indicates at any given time the number of pieces of each asset prototype that are available at the HQ location. Engineer staff units are simply HQ units with certain characteristics (namely, their GM.UN.SUPERIOR is not an engineer).

The engineer work unit is a mechanism to move the engineer assets around the battlefield in a way that makes them visible as GG.UNITs to the other modules in the simulation. Work units are created internally, and their attributes are determined by their current use. A work unit is filed in its side's EN.SET.OF.FREE.UNITS when it is not active. When an HQ unit is ready to organize equipment to send to a job, a work unit is removed from the free set and filed in the HQ unit's set of performing units. The necessary assets are transferred to the work unit, which has its own inventory to keep a record of its assets so that they may be properly returned when the work is finished. In the course of processing the jobs on its list, a work unit may own assets which do not belong to its HQ unit and may have some of its assets preempted before its jobs are complete.

A work unit can have one of three possible destinations: the location of the job site for the first job on its list, a rendezvous location for itself and other work units assigned to the same job phase, or a rendezvous path point with its HQ unit. When its purpose has been fulfilled, the work unit transfers its assets to another unit (either another work unit which will move the assets to a new job or back to base or the HQ unit which owns the assets held by this work unit) and becomes inactive again. It is removed from the set of performing units and refiled in the free set.

Engineer Command Structure and Decisionmaking

Each ground unit, including engineer units, has a tactical superior, a list of subordinates, and a tactical area. The unit's tactical superior indicates where the unit fits in the Army organizational structure and is set by the command structure data in the GM input data. Each engineer unit must be in one of two categories:

1. Has an engineer tactical superior
2. Has a nonengineer tactical superior.

Engineer units in the first category are in a command chain of engineer units which ultimately reaches a unit in the second category. Engineer units in the second category are engineer staff units. The types of jobs dynamically generated for this unit and its subordinates and the priorities attached to those jobs are determined by the activities of its nonengineer superior and that unit's nonengineer subordinates. In particular, job responsibility is first determined by looking at the staff units, and all jobs assigned to a staff unit and its subordinates are prioritized and processed from one combined list. This is because all engineer units under a given staff unit share equipment, perhaps combining equipment from several engineer units to accomplish one job.

A list of the engineer staff units in the unit data base is constructed during the initialization process. This list is ranked by the level of the associated ground units, with the lowest echelon units first. In the

search for a job's responsible unit, engineer HQ units in the requesting unit's chain of command are considered first. If no supporting engineer is capable of the work or if no requesting unit was specified for the mission, all staff units are checked for the assignment, with the lowest level staff units being considered first. Each staff unit keeps an array of the techniques for which the needed equipment is owned by its command chain so a quick determination can be made as to whether or not this unit or one of its subordinates will be able to do a specific job. Because a staff unit and all of its subordinates are able to share equipment, the staff unit prioritizes the job list of its entire command chain and controls the job processing sequence to ensure that the most important jobs are activated first.

All units have tactical areas, i.e., portions of the battlefield for which they are responsible. The tactical areas of units must be nested to agree with the tactical command structure. An engineer unit may be placed in general support of a region by the appropriate specification of its tactical area and superiors. An engineer unit with an engineer tactical superior will work within its assigned tactical area on jobs generated by the activities of the units in its area which are in the command chain of its staff engineer's superior.

Engineer Assets

Engineer assets are originally defined in the FL data as weapon prototypes. This is because a unit's mass, attrition, and supply needs are determined by its weapon assets. The FL data also sets the number of each type of asset held by each ground unit, including engineer HQ units. The engineer asset prototype data in the EN data specifies those weapon prototypes which are engineer assets and adds more information about the engineer assets being modeled. Nonengineer units may own weapons identified as engineer asset types, and engineer units may own weapons not identified as engineer asset types.

The engineer assets are actually identified in the EN data, but each engineer asset must correspond to a specific weapon prototype of the same name in the FL data. The EN.ASSET.PR.WP.PTR identifies that weapon prototype. Each of these prototypes may be a specific equipment item (e.g., M9 ACE), a generic equipment item (e.g., dump truck), or an aggregate (e.g., bridge team). An engineer data set may contain a combination of specific equipment prototypes, generic prototypes, and aggregate prototypes.

To restrict the continuous use of assets, each asset has a data-specified maximum work period before rest and a data-specified length of required rest period. Each time an asset returns from a work assignment, its inventory record is updated. When its hours worked exceeds the maximum work period for its prototype, the asset is made unavailable for assignment for the specified rest period.

When the EN module data is loaded, an inventory is constructed for each HQ unit. This list has a record for each piece of engineer equipment owned by the unit. The HQ unit inventory is ordered by asset prototype so that all items of the same type are listed together. When an asset is assigned to a work team, that asset is removed from the inventory and refiled in the inventory at the bottom of its prototype section. This method keeps the most available assets at the top of the respective sections, thus allowing quick location of assets available for a new job.

A unit's inventory list grows or shrinks only when a formal transfer of an asset is made from or to another HQ unit. When a work unit has control of the asset, the asset's record is still with the parent unit, but with appropriate attributes set to indicate the asset's status. Monitoring this inventory list yields engineer output data regarding the use of assets.

Some engineer assets (e.g., armored vehicle launched bridges [AVLBs]) may not return with the work unit when a job is complete, though the asset still exists and can be reused if retrieved. The job technique information indicates which equipment fits this description. When such a job is completed, the work unit returns to its HQ unit without the retrievable equipment. In the current implementation of the EN module, only assets used for breaching a line obstacle may be retrieved. When an explicit line obstacle breach using retrievable assets is completed, an automatic mission is generated to improve the breach site. If this mission is completed, then the retrievable assets are returned to the owning HQ unit. The external event EN.BRIDGE.DEMOLITION may also be used to retrieve assets. If a side destroys a bridge which was constructed with retrievable assets owned by that side, then the bridge disappears and the retrievable assets are recovered. If the enemy destroys such a bridge, then the bridge disappears and the assets are destroyed.

Engineer Techniques

While an engineer job is determined by the task type, feature type, and location, the engineer effort to get the job done is determined by the technique chosen according to the equipment and supplies available. For each side and task type/feature type, input data specifies a set of techniques by which the task may be done. The order in which the techniques are listed in the input data determines the order in which the techniques will be considered for use, with the most preferred technique listed first.

Each technique is associated with a command level that identifies the echelon normally tasked with this type of work. In searching for a responsible unit, an attempt will be made to assign a job using this technique to the unit closest to this level in the appropriate command chain.

Each technique has associated with it a minimum signed distance from the forward edge of battle area (FEBA) which is used to distinguish those techniques that would be used in an assault situation from those used in rear areas. If the job site is closer to the FEBA than this minimum value, the technique will not be used. A positive sign indicates a distance from the FEBA on the friendly side, while a negative sign indicates a distance from the FEBA on the enemy side. A technique which could be used without regard to FEBA location should have a negative sign and a relatively large absolute value, e.g., -250 km. A technique that would be used only in the rear would have a positive sign and an absolute value indicating how far away the nearest enemy should be, e.g., 30 km.

Each technique has a point of organization, which is at either the responsible HQ or the job site. If a job is using a technique with a HQ organization point, then all of the equipment needed for the current phase will move first to the responsible HQ unit's location and form one work unit to proceed to the job site. If a job is using a technique with the job site organization point, then each set of collocated equipment being used for the job moves as a unit directly to the job site, forming one work unit there. Work begins as soon as the equipment required for the current segment is at the job site. In addition, when the organization point is at the job site, equipment leaves for its next destination as soon as it is no longer needed at its present job site. When the organization point is the HQ unit, the equipment working on a phase of the job stays together until the phase is complete.

Each technique has a relative effect associated with it which is intended to distinguish between the effects of different types of engineer techniques on the same feature type. For example, all bridges do not have the same delay factor for crossing units. The feature type has associated with it a standard breached delay. The breached delay associated with a particular engineer breach is computed by dividing the standard delay by the relative effect. For example, if the breached delay for a given line obstacle is specified to be 20 minutes in the TB data, a bridge with a "relative effect" of 2 would have a delay of 10

minutes. See Chapter 3, **INPUT DATA DESCRIPTIONS** (p 31), for a complete description of the use of relative effect.

The key structure for tracking a job through many steps is the job segment. The segments mark discrete points at which an effect can be measured and a segment duration time can be predicted. The segments may be a part of a continuous work effort that ties up all needed equipment for the duration or a disjoint set of activities using different equipment in phases that only tie up the equipment used in that phase. A phase is defined to be from the current segment to the end of the first succeeding segment with an end action that is a break point or end of job. Dividing a job into segments allows for a job to be preempted by a higher priority job and for work units to work in the assault, where fluctuations in available equipment alter the duration or cancel a job. Also, equipment used only at the beginning of the technique may be released for other jobs before the present job is finished, and interrupted jobs can have a measured effect and can be resumed without starting from the beginning. When a segment is completed, the EN.END.OF.SEGMENT.EFFECT specifies the fractional engineer effect of the work to this point. If work on a segment is interrupted after it has begun, the intrasegment effect is used.

Each technique has a list of required equipment that determines the type of equipment to be used and the preferred and minimum number of pieces of each type. This list is used first when it is compared with a unit's assets to determine whether or not the unit can do a job by this technique, with the criterion being that the unit own the minimum required number of each type of equipment. If this technique is chosen, this list is used to detect when an asset is needed at the job site and to determine job segment duration after assets have been damaged. The segment duration is based on the availability of the preferred amount of equipment. If less than the preferred amount of equipment is available, the duration is increased proportionally. If equipment levels fall below the minimum required, the mission is discontinued.

When equipment is damaged or preempted, the time to complete the current and future segments will be adjusted to reflect the loss of the equipment. If only one type of equipment is lost, completion time for a segment will be adjusted to reflect the capacity of the current number based on the duration and capacity of the original number. If equipment of different types is lost, the previous computation will be done for each type and the maximum duration time used.

The base preparation time for a technique's required equipment is used to determine pre-job activity time for the work unit. If any of the equipment requires base preparation time, the maximum of those amounts is used to determine the departure time.

Engineers and Logistics

When VIC's logistics option is selected, engineers have requirements for two kinds of supplies: equipment and job supplies. The supply inventory of the HQ unit owning the equipment being sent to a job is checked for that equipment's ammunition and fuel requirements. The supply inventory of the HQ unit responsible for the job is checked for the required job supplies.

When a piece of equipment is transferred to a work team, a transfer of the fuel and ammunition required for that equipment takes place. If the amount on hand is less than the amount needed, a report is sent to the engineer history file and the supply level is artificially raised to handle the deficit.

This is not the case for job supplies. A job is not processed until the responsible HQ unit has the supplies required for the job in its inventory. The logistics module links supplies to weapon types, with

the type and amounts of supplies delivered to a unit dependent on the weapons that unit owns. For engineer job supplies, no corresponding weapon usually exists. An artificial weapon type may be created to handle this difficulty. The supply needs for this weapon type would be equal to the job supply requirements for the owning HQ unit.

The Engineer Task Performance Sequence

The engineer task performance sequence was simulated explicitly to make the timing of the task completion sensitive to the factors that logically affect engineer capability. Those factors arise from the input descriptions of force structure, resources, and missions and from the evolving situation as the simulated battle progresses. The only way to take them into account completely is to simulate the process explicitly. However, sometimes a rough estimate of capability using less processing time would be good enough, so a second estimating method—the implicit representation—was added. This method is still sensitive to many of the same factors as the explicit representation, essentially being a "best case" estimate of the completion time.

Whether the representation is implicit or explicit, the task performance sequence is the same. The distinctions between the two representations occur in the way each step of the sequence is accomplished. Those distinctions are described below.

The six steps of the engineer task performance sequence are:

1. Recognize the need to perform the task

This step is the same for both the explicit and the implicit representation. Each side builds its mission list as the need for engineer activity arises. The mission lists are periodically processed to determine who should be assigned the task and how the task should be done.

Each type of mission has its own method of generation, as indicated in Figure 1. Note that, except for position preparation, the task generated through input data may or may not have a specified requesting unit. In order for such a task to be done implicitly, a requesting unit must be specified.

In order to make engineers more responsive in their mobility function, a planning process was added which scans the paths of each maneuver unit for encounters with known obstacles and generates the mission to breach before the unit actually arrives at the obstacle.

2. Assign the task to a unit and determine how the task is to be done

a. Periodically an attempt is made to assign each mission on the side's mission list. The mission owns a list of units capable of doing the work. If that list is empty, which is the case the first time a mission is processed, the procedure moves to step b, otherwise to step c.

b. If the mission has a requestor, every engineer staff unit in the requestor's command chain is checked and, if capable, put on the mission's possible unit list. A unit is capable if that unit and its subordinates collectively own the minimum equipment required for a technique by which the mission may be done and if the staff unit has a path point with a tactical area containing the work site. In addition, the requestor's engineer assets are checked to see if the requestor is capable of doing the work implicitly. If capable, the requestor is added to the list of possible units, and a unit attribute is set to trigger periodic checking for the work to start. If no requestor has been specified for the mission, the entire set of staff

TASK	GENERATION METHOD	REQUESTOR
Breach a minefield	Unit encounter	The unit itself
Breach a line obstacle	Unit encounter	The unit itself
Breach an obstacle complex	Unit encounter	The unit itself
Construct a combat trail	TB input data	As specified
Emplace a minefield	MF input data	As specified
Emplace a line obstacle	TB input data	As specified
Emplace an obstacle complex	MF input data	As specified
Destroy a line obstacle breach	TB input data	Supreme unit and external event
Prepare a protective position	GG input data	Unit having the given path point
Clear a minefield	Unit encounter	The unit itself
Improve a line breach	Enemy destruction of an MSR bridge and retrieval of assault bridging	Supreme unit or unit completing assault bridge
Maintain a road	Road use at data threshold	Supreme unit

Figure 1. Engineer Mission Generation.

units for the side is checked and every capable staff unit is added to the possible units list. If this process ends with the list of possible units still empty, then the mission is deemed impossible and is canceled.

c. If the early start time for the mission will occur before the next mission processing interval, then continue with step d. Otherwise, the mission stays on the mission list, and nothing more is done to the mission in this cycle.

d. Each engineer staff unit on the list of possible units is checked to see if the unit's current tactical area contains the mission site. If no units currently have the site in their tactical area, nothing more is done to the mission in this cycle. If only one such unit is found, then that unit's command chain will be assigned the task. If more than one unit is found, then the unit currently assigned the least number of jobs is chosen. If two such units have the same number of jobs, the unit closest to the job site is chosen.

e. If an engineer staff unit is chosen in step d, then the task performance will be explicitly simulated. The next step is to choose the appropriate engineer HQ unit subordinate to the chosen staff. The technique to be used was decided in the process of choosing the staff unit. Now the search narrows to the subordinate closest to the job site, owning the most equipment required by the technique, and at an echelon closest to the technique's normal echelon. The mission is removed from the mission list, and jobs are created for every feature involved with the mission. The jobs are placed on the chosen HQ unit's pending job list, and an event to prompt the staff unit to process the job lists is scheduled.

f. If no engineer staff unit is chosen in step d but the requesting unit is in the list of possible units for the mission, then a check is made to determine if the requesting unit currently has the mission location within its radius of control. If so, the task will be done implicitly by the requesting unit.

NOTE: When a mission is assigned to an engineer HQ unit, then the task performance and all of the engineer processes associated with it are explicitly simulated. The HQ unit's staff prioritizes all jobs assigned to his command chain and attempts to activate each of the pending jobs in priority order. The staff unit is responsible for deciding the order in which jobs assigned to units in his command chain are processed. This is because the units in a staff's command chain can pool equipment. Each time the staff unit processes the job list, a new priority ranking is established for all jobs, pending and active. If a pending job moves above an active job in ranking, then the pending job may preempt equipment from the active job. The active job may then require a longer completion time or may be canceled if the equipment levels drop below minimum requirements.

Each time a nonengineer unit arrives at one of its path points, its implicit engineer mission indicator is checked. If the indicator is on, then the mission list for the unit's side is checked for missions with this unit in the possible units list. If no missions are found, then the indicator is turned off. Otherwise, a list is made of all such missions currently within the unit's radius of control. The list is arranged in priority order according to a ranking system which considers the unit's current combat status, the type of task involved, the margin of time between the earliest possible completion time and the latest acceptable completion time, and the distance of the task site down the unit's path from its present location. The missions are processed in priority order. If the unit has the equipment and supplies available to do the task, then an estimate of the completion time is made from the technique data using current conditions and ideal travel time. With a predicted completion time in hand, the mission is removed from the mission list, and its completion is scheduled for the computed time. The process moves to the next mission on the list, keeping track of equipment commitments as it goes.

3. Check that the required equipment and supplies are available

For the explicit representation:

a. The responsible HQ unit looks for equipment needed for the current phase of the job in this order:

- belonging to the HQ unit and available
- belonging to HQ units subordinate to the current one and available
- belonging to the HQ unit superior to the current one (and to its other subordinates) and available
- belonging to the HQ unit and working on a lower priority job (considered in priority sequence)
- belonging to HQ units subordinate to the current one and working on a lower priority job
- belonging to the HQ unit superior to the current one (and to its subordinates) and working on a lower priority job.

b. If the minimum amount of the needed equipment can be found and the responsible HQ unit has the required supplies on hand, the equipment is assembled by transferring each block of found equipment to a work unit and sending the resulting work units to a rendezvous point as determined by the technique's organization point. The search for equipment continues until the desired amount is found, so equipment levels actually used may range between the minimum and preferred amounts. Each asset keeps a record of who owns it so that it can return to that HQ unit when the phase is finished. Assets being taken from a lower priority job leave immediately, and the lower priority job increases its work time or discontinues according to whether or not minimum equipment levels are left.

4. Determine how long it will take to move the equipment and supplies to the site and perform the task

a. All of the work units join into one work unit at the rendezvous point and, if necessary, proceed to the job site, with appropriate delays for base preparation. Each work unit picks a path to the work site which avoids as many known obstacles as possible. Work units are moved by the same processes that are used for all other ground units in VIC. The work unit has a list of jobs which are to be performed before it disbands. This list is constructed at the time the original work units are being formed. Jobs on the responsible HQ unit's pending jobs list which have the same mission number are added to the work unit's job list.

b. The job phase currently being processed is done one segment at a time according to the segment information associated with the technique chosen for this job. For a technique with organization point at the job site, equipment which is no longer needed at the end of a segment is assembled into a new work unit and sent to the next job on the list or to its owning HQ unit if all jobs on the list have been done.

c. When a work unit is ready to begin the next segment of its current job, a comparison is made between the equipment levels of the work unit and the desired equipment levels for the technique being used. If the levels agree, then the base segment duration is the data-specified duration. If the equipment level is between the minimum and the preferred amount, the base segment duration is adjusted proportionally. Then the base segment duration is adjusted for weather, day/night, and combat conditions, and the end of the segment is scheduled for the computed time.

d. Work on the job continues from one segment to the next, with job attributes keeping track of how much has been done. When the end effect of a segment is 1.0, the work is complete.

5. Expend the required supplies and make the equipment unavailable for use on another task until the current task is complete

a. The technique supply data specifies the segment during which supplies are expended. At the end of the segment, the supply inventory of the work unit is adjusted to reflect that the supplies are no longer available. Until that time, if work is interrupted, the supplies return with the work unit to the responsible HQ.

b. The equipment required for the current job is not available for use on other jobs unless it is preempted. This mechanism automatically takes care of ensuring that the amount of work accomplished is commensurate with the level of equipment.

6. Register the effect of work completion at the end of the calculated work time.

a. When the work on a job is complete or the job is interrupted at a point where the level of the end effect can be measured, the appropriate adjustment is made in the physical feature involved with the task. Both the explicit and the implicit representations use the same routine to do this.

b. When the task performance is explicitly represented, the work unit must travel back to the HQ unit to return equipment. However, the equipment is available for new assignment as soon as the work on its current job is complete.

3 INPUT DATA DESCRIPTIONS

Introduction

Coordination of the input data of several modules of VIC is necessary to take full advantage of the capabilities offered by the new engineer module. The EN data focuses primarily on task performance specifications, stating how engineer tasks are done and what equipment and supplies are required to do them. The GG, GM, FL, and LO data modules describe the engineer units, their organizational structure, and their equipment and supplies. For implicit engineer play, the resources allocated to maneuver units in the FL and LO data modules, in conjunction with the EN data, determine the abilities of such units to perform engineer tasks. The terrain features—minefields, line obstacles, obstacle complexes, defensive positions, roads, and combat trails—which are emplaced, removed, breached, or maintained by engineers, are described by both type and occurrence in the MF, TB, GG, LO, and EN modules.

The functioning of the EN module—the who, what, where, when, and how of task performance—also depends on and is very sensitive to certain data in the aforementioned data modules. In the GG data, the unit paths and tactical areas of engineers and their supported units affect mission assignment and work team travel distances. The position preparation times for unit prototypes and the exposure rates for each unit combat status combine with the engineer technique data for position preparation and should be consistent with it. In the GM data, the unit command structure affects how missions are assigned and how engineer equipment is pooled among several headquarters units. In the FL data, weapon prototype information determines the vulnerability of engineer equipment to direct and indirect fire, thereby affecting the accuracy of the representation of performance capabilities in combat. In summary, the GG, GM, and FL data modules play a crucial role in determining what engineer tasks the units in the scenario will be capable of doing.

The TB, MF, LO, and GG modules contain the information that specifies what engineer tasks need to be done: GG unit path data generates position preparation missions; TB data generates combat trail construction and line obstacle emplacement and determines the unit delays which affect the generation of line obstacle breaching missions; MF data generates minefield and obstacle complex emplacement and determines the unit delays which affect the generation of minefield and obstacle complex breaching missions; and LO data generates MSR bridge and road data which affect the generation of bridge improvement and road maintenance missions.

VIC has no built-in data checks to ensure that interdependent data, spread across several modules, is consistent or even logical. The scenario builder is left with that task. The effort will be eased if the scenario builder is familiar with this report's **METHODOLOGY** chapter (p 7), which explains the structure of the EN module and how it uses the data from those other key modules. The data items mentioned above were not added to accommodate the EN module. Except for the new terrain features (combat trails and obstacle complexes), the data in question has always been required for a VIC scenario. The new EN module simply uses more of the scenario data than previous versions to portray engineer processes more accurately. Indeed, outside of the EN data module, the new EN module requires only two new data items in the GG module to handle a change in the engineer unit structure.

The EN module has two types of engineer units: the HQ unit and the work team. In the current version, the engineer headquarters units are very flexible entities. As ground units, they have no special restrictions placed on them regarding what they may own, how they move and fight, and where they fit into the force structure. In addition, HQ unit prototypes may be at any level except platoon, which is

reserved for the work team prototype. The prototype specifications for GG units must contain at least one prototype with type engineer (ENG) and level platoon (PLT) to describe the engineer work team. The first such prototype listed in the ground unit prototype data of GG is assumed to be the BLUE engineer work team prototype. The second such prototype, if it exists, is assumed to be the RED engineer work team prototype. If the prototype data contains only one prototype of type engineer and level platoon, then this is assumed to be the RED engineer work team prototype as well. The unit data base of the GG data may not contain any units with the engineer work team prototype as their prototype specification.

The new engineer module simplifies the engineer unit structure in VIC by removing the work teams from the GG unit data. Unlike earlier versions of VIC in which all of the work teams and their attributes were specified in the input data, the new engineer module "creates" the work teams internally as they are needed and task-organizes them for a specific set of jobs. Only the engineer headquarters units are identified in the GG data. Consequently, headquarters units are the only engineer units appearing in the unit data sections of GM, FL, and LO. The headquarters units hold the only resources (equipment and supplies) available to engineers for the explicit simulation of task performance. They use their equipment and supplies for task performance by transferring what is needed to a newly-created work team and sending the work team to the site. The EN data determines what is needed and how long the job will take.

To be able to "create" work teams, which must be GG.UNITs if they are to be visible to the rest of VIC, the model must create two pools of extra GG.UNITs, one for each side. The free GG.UNITs have their attributes set to make them invisible when they are not in use. "Creating" a work team simply means setting the attributes of an unused GG.UNIT so that it is visible and sending it on its way. After the work team has served its purpose, it returns to its invisible state to await being used again. This process allows the new version of VIC to accomplish more for engineers without increasing the number of GG.UNITs dramatically. Two numbers specifying how many extra BLUE and RED units to create are new data items in the GG module, being input immediately after the number of BLUE and RED units in the data base. Determining the number of extra units to create is not easy.

The number of extra GG.UNITs needed to simulate engineer task performance depends on many scenario variables: the total number of ground units; the level of detail of the engineer asset data; the complexity of the engineer technique data; the number of preplanned obstacles, minefields, and defensive positions; the number of natural obstacles in the terrain, etc. The engineer module cannot function if it depletes either of the sets of extra units. On the other hand, the run time will be affected if too many extra units are created. To aid in optimizing these two numbers, the minimum number of extra BLUE and RED units available during a run is written as the last line of the engineer setup file, **ENSTP.LIS, at the end of each model run. The numbers of work units in the GG data may be reduced by these values for subsequent runs of the same scenario.

The documentation in the *Methodology and Input Data Manual* describes the engineer-related data in the GG, GM, FL, LO, MF, and TB data modules. The **METHODOLOGY** chapter of this report explains how that data affects the processes of the engineer module. The input data required for the EN data module is described below, divided into seven segments:

Segment EN-ONE	Engineer Timing Data
Segment EN-TWO	Defensive Position Data
Segment EN-THREE	Engineer Job Priority Data
Segment EN-FOUR	Engineer Asset Prototype Data
Segment EN-FIVE	Engineer Technique Data

Segment EN-SIX	Required Equipment For Engineer Techniques
Segment EN-SEVEN	Required Supplies For Engineer Techniques

Engineer Timing Data

The first items in the EN data module control the timing of the main engineer functions. For each side in the simulation, four times in day/hour/minute format are required:

MISSION PROCESSING INTERVAL - the maximum length of time allowed between successive passes through the side's mission list in an attempt to assign missions. Each side maintains a list of its unassigned engineer missions. When the list is processed, all missions that can be assigned are removed from the mission list and converted to engineer jobs. If this processing interval is short and the mission list contains a large number of missions which cannot be assigned because of current conditions, the run time may be affected. On the other hand, the larger this interval is, the less responsive engineers will be to requests for support. Dynamically generated mobility missions are processed immediately and do not use this interval.

ENGINEER PLANNING INTERVAL - the maximum amount of time allowed between a unit's request to breach a known obstacle in its path and its actual encounter with that obstacle. Each unit scans its path for known obstacles and creates a planning path point for each. A path point arrival at the planning point generates an engineer mission to breach the obstacle. The planning path point is positioned in the path to allow as much time as possible, up to a maximum of the planning interval, for the unit to request engineer assistance. With explicit engineer play, work teams will be sent to the site as soon as the necessary resources are available, since work team movement is independent of the requesting unit's movement. With implicit engineer play, if the only breaching resources belong to the requesting unit itself, work will begin on the breach when the requesting unit arrives at a path point that is within the unit's radius of control of the breach site. The engineer planning process may be deactivated by setting the planning interval to zero.

JOB TURNAROUND INTERVAL - the amount of time allowed to elapse after equipment or new supplies become available or new jobs are assigned before the engineer unit job list is processed again. The job turnaround interval represents the time required to process equipment and supplies in and out, given that planning and scheduling have already been accomplished. In the case of a new job being assigned to the staff, the assumption is that the staff was adequately forewarned or that the task is one that is normally handled spontaneously. Note that each type of equipment already accounts for a preparation time before leaving its base with a new assignment, so preparation time should not be considered in the job turnaround time. This interval must not be zero, since this would produce an infinite loop in the timing routine.

NEW ROLE INTERVAL - the amount of time allowed to elapse before the pending job list is processed again after a supported unit changes its role by reversing direction on its path. The new role interval represents the time required by staff to assess a changed situation and to plan the supporting engineer activities, including reprioritizing its active and pending jobs and perhaps canceling jobs already in progress. This interval must not be zero, since this would produce an infinite loop in the timing routine.

Defensive Position Data

The defensive position prototype data allows the scenario developer to make distinctions between defensive positions for different types and sizes of units. The structure of the data in this section allows each unit prototype to be linked with a defensive position prototype. That linking determines the engineer techniques that may be used to prepare a position for a unit of a given unit prototype as well as the type of vacant defensive position a unit of a given prototype may occupy. The linking of defensive position prototype to unit prototype attempts to match as closely as possible the unit type and size with the position prototype type and size.

In the simplest representation, only one defensive position prototype may be used. In this case, units of all types and sizes require the same level of engineer effort for position preparation, though the GG data may specify a different self-preparation time for each unit prototype. The resulting positions afford uniform levels of protection from direct fire for all unit types. This case emulates the way VIC represented defensive positions prior to version 3.0.

In a more complex scenario, selected unit types and/or sizes may have specific position prototypes, with all other types and/or sizes of units having a "generic" position prototype. For example, armored units may require specific engineer equipment for the preparation of their positions, and those positions may be unsuitable for any other type of unit to occupy. In this same scenario, all other types of units may prepare and use positions that are so similar that one generic type can be used to represent them all.

In the most detailed representation, every type and size of unit may have its own defensive position prototype. In this case, engineer data for the preparation of positions could be very specific as to time and equipment required, and only units of the same prototype would be allowed to use the same positions.

The data required for defensive position prototypes is:

NUMBER OF DEFENSIVE POSITION PROTOTYPES - the number of defensive position types to be defined in this data. This number of defensive position prototypes will be described below.

For each defensive position prototype, specify:

PROTOTYPE NAME - the name for the prototype, to be used to link position types with the engineer technique data and to identify position types in output reports.

UNIT TYPE - the type of unit this defensive position is designed to protect. This may be any character string. To limit the use of this type of position to a particular type of unit, this item must match the abbreviation of the corresponding GG.UNIT.PROTOTYPE type of the GG data:

"TNK" = TANK UNIT
"MEC" = MECHANIZED UNIT
"INF" = INFANTRY UNIT
"CAV" = CAVALRY UNIT
"AH" = ATTACK HELICOPTER UNIT
"UH" = UTILITY HELICOPTER UNIT
"CH" = CARGO HELICOPTER UNIT
"HQV" = AVIATION HQ UNIT
"HQA" = ARTILLERY HQ UNIT

"HQA" = ARTILLERY HQ UNIT
"TUB" = TUBE ARTILLERY UNIT
"RKT" = ROCKET ARTILLERY UNIT
"SSM" = MISSILE ARTILLERY UNIT
"ADA" = AIR DEFENSE UNIT
"HQ" = HEADQUARTERS UNIT
"SUP" = SUPPLY CONVOY UNIT
"FSA" = FWD SUPPLY AREA
"RDU" = REPAIR UNIT
"ABN" = AIRBORNE UNIT
"ENG" = ENGINEER UNIT
"INT" = INTELLIGENCE UNIT
"JAM" = JAMMING UNIT
"EH" = ELECTRONIC HELICOPTER UNIT

If the unit type does not match any of the above, then this position will be deemed suitable for all types of units.

UNIT LEVEL - the level of unit this defensive position is designed to protect. This may be any character string. To limit the use of this type of position to units at no higher than a certain level, this item must match the abbreviation of the corresponding GG.UNIT.PROTOTYPE level of the GG data:

'FRT' = FRONT
'ARM' = ARMY
'COR' = CORPS
'DIV' = DIVISION
'RGT' = REGIMENT
'BDE' = BRIGADE
'BN' = BATTALION
'TF' = TASK FORCE
'CO' = COMPANY
'SQN' = SQUADRON
'TRP' = TROOP
'BTY' = BATTERY
'PLT' = PLATOON

If the unit level does not match any of the above, then this position will be deemed suitable for all levels of units.

Engineer Job Priority Data

The priority for an individual job is a real number derived from several contributing factors, including the task type, the combat status of the requesting unit, the distance of the requestor from the task site if it is a mobility or survivability task, the margin of time between the earliest possible finish time and the latest acceptable completion time for the mission, the difference in level between the requestor and the responsible HQ, and whether or not the earliest start time of the mission has passed. The task type combined with the combat status of the requesting unit establish the beginning of the priority range for each task by using the table provided in this section of input data. Each combat status has an

associated mapping to one of three modes: attack, defend, withdraw; and each engineer task has an associated mission type: mobility, countermobility, survivability, and general engineering. The table defined by the data below establishes the relative importance of each mission type in each combat status mode.

For each side, for each combat status mapping, and for each engineer mission type, specify:

RELATIVE PRIORITY - an integer (1 to 4) giving the relative ranking of the engineer mission type in the given combat status mode, with the larger number indicating a higher priority. The model will accept any integer value since this number is only used as a multiplier. In particular, a zero may be used to push certain task types to the bottom of the priority ranking, but the same ranking could be accomplished with the proper use of the numbers from 1 to 4. Engineers will attempt to do every mission generated; this data does not offer the possibility of totally stopping certain types of tasks for particular combat status mappings.

Engineer Asset Prototype Data

The engineer asset prototype data identifies those weapon prototypes from the FL data which are required to perform engineer tasks and provides additional information concerning capabilities and limitations.

The data required for the engineer asset prototypes specifies:

NUMBER OF ENGINEER ASSETS - the number of different types of engineer assets being modeled. Assets prototypes are first specified in the FL input data as weapon prototypes. This is the number of such weapon prototypes which are to be identified as engineer asset prototypes below.

For each engineer asset prototype:

ASSET NAME - the name by which the corresponding weapon prototype was identified in the FL data. Each name here must match one of the weapon prototypes. This data item establishes the link between the asset prototype's weapon data and the asset prototype's engineer data.

ASSET SPEED - the basic movement rate of this asset prototype across open country in kilometers per hour. The speed at which an engineer work team moves will not exceed the minimum value of this speed for the assets owned by the work team.

ASSET DAMAGE THRESHOLD - a real number between 0.0 and 1.0 indicating the minimum portion of an asset of this type which must be functional in order for the asset to continue working. An asset of this type which has suffered attrition continues to work at a degraded level until its undamaged portion falls below this number. At that point, the asset is declared ..DAMAGED and is no longer available for assignment.

ASSET MAXIMUM WORK PERIOD - the maximum amount of time in hours that this asset can work continuously without a rest period.

ASSET REST PERIOD - the amount of rest time in hours required by an asset of this type after a maximum work period.

ASSET REPORT FLAG - a "Y" (yes) or "N" (no) to indicate whether or not a detailed asset report is to be generated during the run for assets of this type. If "Y" is chosen, the model will print an asset report line to the engineer history file every time the status of an asset of this type changes. If "N" is chosen, no asset reports will be generated for assets of this type. If engineer assets are being modeled in detail with many engineer HQ units owning many assets of this type, the run time will be severely affected when the asset report flag is set to "Y".

Engineer Technique Data

This section of data allows the scenario builder to identify the different methods that engineers may use to accomplish a specific task, as determined by task type and feature type. The techniques may differ in a number of ways:

1. the effect of the end result
2. the equipment or supplies required to accomplish the work
3. the time required to accomplish the work
4. the work pattern established for the efficient use of the equipment
5. the suitability of the technique for the particular situation.

For example, if the task is "breach a linear obstacle" and the feature type is "50M River," then two different bridging techniques may:

1. produce bridges that have different capacities
2. use different engineer assets in the work stages
3. require different amounts of time to complete
4. differ only in the way in which the work is organized
5. identify one technique for assault locations and another for rear area locations.

If more than one technique is listed for a given combination of side, task, and feature, then those techniques must be entered in the order in which they are to be considered for use. In most cases, the first technique in the list with attributes matching a particular mission's requirements will be chosen to accomplish the mission.

NUMBER OF TECHNIQUES - the total number of techniques to be specified in the data below for all of the task types and feature types. The engineer task types are identified below. The feature types are specified in other VIC data modules: minefield prototypes and obstacle complex prototypes in MF, linear obstacle prototypes and combat trail vegetation types in TB, defensive position prototypes in EN, and road prototypes in LO.

For each technique:

TECHNIQUE NUMBER - a sequential numbering of the techniques to give a reference number to the technique for the scenario builder and for the simulation.

TECHNIQUE SIDE - the side ("B" for BLUE or "R" for RED) that uses this technique.

ENGINEER TASK NUMBER - the task type of the given technique. The engineer tasks will be associated with the following numbers:

..MINEFIELD.BREACH	= 1
..MINEFIELD.CLEAR	= 2
..LINE.OBSTACLE.BREACH	= 3
..BREACH.OBSTACLE.COMPLEX	= 4
..IMPROVE.LINE.BREACH	= 5
..REPAIR.ROAD.CRATER	= 6
..BUILD.TRAIL	= 7
..EMPLACE.MINEFIELD	= 8
..EMPLACE.LINE.OBSTACLE	= 9
..EMPLACE.OBSTACLE.COMPLEX	= 10
..BRIDGE.DEMOLITION	= 11
..PREPARE.POSITION	= 12
..CRATER.ROAD	= 13
..MAINTAIN.ROAD	= 14

FEATURE PROTOTYPE NAME - the name associated with the feature type for this technique from the data module where the prototype description is given. For minefields, this is the mine name from the mine type input data in the MF module. For obstacle complexes, this is the obstacle complex prototype name in the MF data. For linear obstacles, this is the obstacle name from the line obstacle input data section of the TB module. For defensive positions, this is the name of the defensive position prototype in the EN data. For combat trails, this is the vegetation symbol used in the TB input data to describe the ground cover in a grid. For roads, this is the road prototype name from the LO road data.

NORMAL COMMAND LEVEL - the echelon at which this type of technique is normally employed to do the task. In the search for a responsible unit for a job using this technique, an attempt will be made to assign the job to a unit at this level. The following text identifiers are used to designate the command level:

'FRT'	= FRONT
'ARM'	= ARMY
'COR'	= CORPS
'DIV'	= DIVISION
'RGT'	= REGIMENT
'BDE'	= BRIGADE
'BN'	= BATTALION
'TF'	= TASK FORCE
'CO'	= COMPANY
'SQN'	= SQUADRON
'TRP'	= TROOP
'BTY'	= BATTERY

Engineer work teams are assigned the level PLT. No engineer units of that level may be in the GG unit data base, and PLT may not be used as a level for this data entry.

MINIMUM DISTANCE FROM FEBA - the minimum distance in kilometers from the FEBA at which this technique will be used. This allows the scenario developer to distinguish techniques for assault situations from techniques used only in the rear. This represents the signed distance from the FEBA for a side, with negative numbers indicating the distance from the FEBA on the enemy's side.

RELATIVE EFFECT - a positive real number used to vary the end results of different techniques for the same task and feature type. This applies only to certain task types. The value of this data item and its use differs for each applicable task type; for all other task types this item should be 1. The applicable task types are:

- LINE OBSTACLE BREACH and IMPROVE LINE OBSTACLE BREACH - Each linear obstacle prototype has a standard breach delay associated with it in the line obstacle prototype data. To distinguish between different breach technique results, the relative effect allows tailoring the delay time to this type of breach by dividing the standard breach delay by the relative effect. The scenario developer must correlate the data in the TB module with the engineer data. For tasks involving line obstacle breaches, the relative effect may be any positive real number.

- MINEFIELD BREACH - The minefield data specifies a clearance factor for minefields that have been breached. This number between 0.0 and 1.0 represents the portion of mines cleared by a maneuver unit breaching without engineer assistance. If either implicit or explicit engineer activity is responsible for the breach, then the relative effect of the technique used for the breach is a multiplier of the clearance factor.

- POSITION PREPARATION - For defensive positions, different techniques for the same position prototype will produce equal protection from direct fire but may give different protection from indirect fire. In this case, a relative effect between 0.0 and 1.0 is allowed. With a relative effect of 0.0, the position would give no better protection from indirect fire than an unprepared position. A relative effect of 1.0 gives maximum possible protection from indirect fire as determined in the exposure mappings of the GG combat status data. Values between 0.0 and 1.0 produce a linear interpolation between the minimum and maximum exposure.

- COMBAT TRAILS - A combat trail raises a grid's trafficability level, which is an integer between 1 (good) and 5 (poor). Different techniques produce different trafficability by using the relative effect to determine the number of levels the trail raises the trafficability. For this task type, the relative effect is rounded to the nearest integer and subtracted from the current trafficability level of the grid to determine the new level.

ORGANIZATION POINT - a flag ("0" for ..AT.HQ and "1" for ..AT.SITE) to indicate how work teams move from their point of formation to the work site. A "0" indicates that when this technique is used, all equipment required for the current phase will rendezvous at the responsible engineer HQ unit and move to the work site as one unit. A "1" indicates that when this technique is used, equipment being used for the job but not currently located at the HQ unit will move directly to the work site and form a single work unit there, with work beginning on the current segment as soon as the equipment required for that segment is at the site. Work units move to succeeding job sites and return to their bases in the same way as they travel to the work site.

NUMBER OF SEGMENTS - the number of distinct, sequential activities required for a job using this technique. Each technique divides a job into segments and phases. A segment is a distinct subsection of the total work effort, perhaps requiring only a subset of the equipment at the site and marking a point at which work might resume if it has been interrupted. A phase is a sequence of segments which are all to be done by one work unit in a continuous effort. This data item specifies the number of segments that the scenario builder chooses for this particular technique. The phases are determined by the end-of-segment action below, with a phase containing all segments from the starting segment until encountering a segment where the end action is a break point or end.

For each segment of the technique:

END-OF-SEGMENT EFFECT - the fractional part of the work's effect at this point in the effort. If the work were interrupted at this point, the effect of the partial work effort can be registered. This is a number between 0.0 (no effect) and 1.0 (total effect complete). Normally each successive end-of-segment effect will be larger than its predecessor.

INTRASEGMENT EFFECT - a pointer to one of four functions which are used to determine the effect of work interrupted in the middle of the segment. The functions are:

1 = NO.CHANGE	(use the effect at start of the phase)
2 = LINEAR	(effect directly proportional to time worked)
3 = SKEW.LEFT	(largest effect at start of segment)
4 = SKEW.RIGHT	(largest effect at end of segment)

SEGMENT DURATION - the length of time in hours needed for the full complement of required equipment to complete the segment. If equipment has been lost or damaged or conditions are worse than normal, this duration is increased. This attribute is used for those segments where completion time is independent of the size of the job. For each segment, if a nonzero duration is specified, then the segment rate must be zero. If the rate is nonzero, then the duration must be zero.

SEGMENT RATE - the rate at which this segment is done if the job has an associated size. For segments whose duration depends on the size of the job, the duration is calculated by dividing the job size by the rate. For minefield emplacement and clearing, the job size is the number of mines and the rate is the number of mines per hour. For linear obstacle emplacement and road maintenance, the job size is the length in kilometers and the rate is in kilometers per hour. For minefield breaching, the size is in kilometers of depth of the minefield and the rate is in kilometers per hour. All other task types are excluded from specifying a segment rate.

VISIBILITY FACTOR - a multiplier of the segment duration which accounts for the change in work time if this segment of the job is done at night.

WEATHER FACTOR - a multiplier of the segment duration which accounts for the change in work time if this segment of the job is done during less than good weather. If the weather is ..FAIR, the segment duration is multiplied by this factor; if the weather is ..POOR, the segment duration is multiplied by the square of this factor.

COMBAT FACTOR - a multiplier of the segment duration which accounts for the change in work time if this segment of the job is done while the unit is within an enemy unit's firing range.

END-OF-SEGMENT ACTION - a flag indicating the action to be taken by the work team after completing this segment. It has three possible values:

1 = CONTINUE, i.e.. go on to the next segment
2 = BREAK POINT, i.e., register the effect, put the remainder of the job back on the pe :ding job list, and move on
3 = END, i.e., the job is complete at the end of this segment.

A job "phase" consists of those segments which have an end-of-segment action of "continue" except for the last segment, which has an end-of-segment action of "break point" or "end". Each new job phase is done by a different work unit with the equipment used only in that phase.

Mark the end of the engineer technique section of the data with "*".

Required Equipment for Engineer Techniques

For each of the techniques identified above, this section lists the engineer assets required for that technique.

TECHNIQUE NUMBER - the number of the technique above for which this equipment is required.

ASSET NAME - the name of the asset prototype of this required equipment. This name must correspond to an asset name in the engineer asset prototype data above.

PREFERRED AMOUNT - the number of pieces of this type of equipment normally used for this technique.

MINIMUM AMOUNT - the minimum number of pieces of this type of equipment required to start or continue work using this technique. Work teams are formed only if the minimum amount of equipment is available, but they are given as much equipment as is available up to the preferred amount. In addition, a work team must have at least this minimum amount of undamaged equipment to work on any segment requiring this equipment.

JOB USE - a flag indicating how the equipment is used at the task site. It may have one of two values:

1 = PERMANENT, i.e., this equipment returns from the work site
2 = RETRIEVABLE, i.e., this equipment stays at the work site but may possibly be retrieved later.

BASE PREPARATION TIME - the time in minutes required to prepare this equipment at its base before it departs for a job site.

SEGMENTS USED - a one line list of segment numbers (separated by spaces) of the segments in which this equipment is used. This item determines the composition of work units working on a job phase as well as the release point for this specific equipment. When the current job segment number is greater than the last segment used for this equipment and the technique's point of organization is at the site, the equipment may be released from the job to go on to another job or return to its headquarters. Mark the end of the list of segments with a "*".

Mark the end of the list of required equipment with a "*".

Required Supplies for Engineer Techniques

For each of the techniques identified above, this section lists the engineer supplies required during the technique. These do not include fuel and ammunition normally carried by units with the above specified equipment but does include supplies like mines or line charges that are used specifically for the task.

TECHNIQUE NUMBER - the number of the technique for which this supply type is required.

SUPPLY NAME - the name of the supply type needed. This name must be identical to a corresponding supply name in the logistics prototype data of the LO module.

REQUIRED AMOUNT - the amount of this type normally used for this technique.

UNIT OF MEASURE FOR SUPPLY - the quantity needed to compute the rate of supply consumption for jobs which have an associated size, allowing the amount of required supplies to be determined by that size. The computation of the total supply requirement uses this formula:

$$\text{Total required supplies} = \frac{\text{job size X required amount}}{\text{unit of measure for supply}}$$

For tasks which may have an associated job size, the unit of measure should agree with the job size unit. For minefield emplacement and clearance, both required amount and unit of measure should be 1 since the job size is the number of mines to emplace or remove. For minefield breaching, using a 150-meter line charge, the required amount should be 1 and the unit of measure should be 0.15. For line obstacle emplacement and road maintenance, the unit of measure will probably be a kilometer.

FIRST SEGMENT NEEDED - the segment number of the technique in which this supply will be consumed. Earlier segments completed by other work units may not have required this type of supply. After this segment, the supply is expended. If work is interrupted before this segment, the supply will be returned.

Mark the end of this list with "*".

Engineer Input Data Format

```
$$$$ EN
$$$  ENGINEER MODULE DATA

'' BLUE       RED
'' -----      -----
                        '' Mission update interval
                        '' Mobility planning time
                        '' Turnaround time between missions
                        '' Time to reschedule activities if new role


''Number of Defensive Positions


''                     Unit   Unit
''  Position Name      Type   Level
'' ---------------     ----   -----


''     BLUE              RED
''---------------  -------------
'' ATK  DEF  WDW   ATK  DEF  WDW
'' ---  ---  ---   ---  ---  ---
                                   ''Mobility
                                   ''Countermobility
                                   ''Survivability
                                   ''General Engineering

''Number of Engineer Assets

''                          Damage  Max Work   Rest    Output
''  Asset  Name     Speed   Thres.   Period   Period   Report
''---------------  ------  ------  --------  ------  --------


''Number of Techniques

''                                        Min.                          Intra-
'' Tech      Task                   Nor  Dist to  Rel  Org  No.  End   Segmnt                       Night  Weather  Combat  EOS
''  No. Side  No.  Feature    Name  Ech   FEBA    Eff  Pt.  Seg  Eff   Effect  Duration  Rate       Factor Factor   Factor  Act
'' ---- ---- ---- ---------------- ---  -------  ---  ---  ---  ----- ------  --------  ------     ------ -------  ------  ---

''Technique Required Equipment

'' Tech                       Pref.    Min.    Job   Prep
''  No.     Asset  Name       Amount   Amount  Use   Time    Segmnts Used
'' ----   ---------------     ------   ------  ---   ----    ------------


''Techniques Required Supplies

'' Tech                       Req.    Unit    Sgmt
''  No.     Supply Name       Amount  Meas    Used
'' ----   ---------------     ------  -----   -----
```

4 OUTPUT

Introduction

VIC's engineer output consists of two files, ***ENGR.LIS and ***ENSTP.LIS, where *** is the prefix chosen for the scenario output files. The file ***ENGR.LIS is a chronological sequence of records of several types written as the simulation progresses. The output lines in this file consist primarily of strings of letters and numbers and are not intended to be used as they are. The engineer postprocessor, ENPOST, uses this file to generate basic reports regarding engineer activity and to create files for standard database management systems for analysis. The second file, ***ENSTP.LIS, which is written at the end of a run, provides the basic setup data for the postprocessor.

While **ENGR.LIS was not designed to be a direct source of information, the user may find some value in sorting this file and using the templates given here to check on a specific activity or resource. Each line in the file ***ENGR.LIS begins with a letter which indicates what type of record it is:

- A Asset record
- B Bridging record
- D Defensive Position record
- J Job record
- L Logistics deficit record
- M Mission (unassigned) record
- N Implicit engineer job record

The templates below are arranged by record type. Each record (line of data) is structured so that applying a standard ASCII sort function will produce a file with recognizable blocks of records containing the chronological history of each asset, bridge, defensive position, etc.

Asset Records

A line is written to ***ENGR.LIS when the status of an asset changes. The sequence of data included on this type of line is:

"A" - the record type

SIDE - the asset's owning side ("1" = BLUE, "2" = RED)

ASSET ID - the asset's "serial number" as determined by the entity's memory address

CURRENT TIME - the time at which the record was written

ASSET STATUS - the new status of the asset:

"1" = Available
"2" = In pretask
"3" = In transit
"4" = Waiting

"5" = Working
"6" = Resting
"7" = Damaged
"8" = Left at site.

TYPE OF ASSET - the index to this asset's prototype as determined by the order in which the engineer asset prototypes are listed in the engineer input file

UNIT NAME - the name of the GG.UNIT presently in possession of this asset

MISSION NUMBER - the mission on which this asset is currently working

JOB NUMBER - the job identifier for the job on which this asset is currently working

UNDAMAGED PORTION - the asset's current FL.WG.STRENGTH

X,Y LOCATION - the asset's current location.

Note that the JOB NUMBER alone is not sufficient to link asset records with job records. The programming language reuses a memory location when a job entity is destroyed and another job entity is created. The combination of JOB NUMBER and MISSION NUMBER, however, uniquely identifies a particular job.

Bridging Records

A line is written to ***ENGR.LIS when a bridge is created, prepared for demolition or demolished. The sequence of data included on this type of line is:

"B" - the record type

SIDE - the side responsible for this change in the bridge status ("0" = NEUTRAL, "1" = BLUE, "2" = RED)

BRIDGE NUMBER - the bridge's "serial number" as determined by the entity's memory address

CURRENT TIME - the time at which the change in status occurred

UNIT NAME - the unit responsible for demolition of the bridge after crossing it

X,Y LOCATION - the x,y coordinates of the bridge

COMPLETION FACTOR - the level of completeness of the bridge, a value between 0.0 and 1.0

BLUE DEMOLITION FLAG - indicator with three possible values:

"0" = unprepared for demolition by BLUE
"1" = prepared for demolition by BLUE
"2" = permission granted to destroy when prepared by BLUE

RED DEMOLITION FLAG - indicator with three possible values:

"0" = unprepared for demolition by RED
"1" = prepared for demolition by RED
"2" = permission granted to destroy when prepared by RED

Defensive Position Records

A line is written to ***ENGR.LIS when the status of the defensive position changes. The sequence of data included on this type of line is:

"D" - the record type

SIDE - the side owning the defensive position ("1" = BLUE, "2" = RED)

DEFENSIVE POSITION ID - the position's "serial number" as determined by the entity's memory address

CURRENT TIME - the time at which the change in status occurred

STATUS - the position's new status:

"1" = position created
"2" = position's completion level updated
"3" = position occupied
"4" = position destroyed
"5" = position provided protection

X,Y LOCATION - the x,y coordinates of the defensive position

COMPLETION FACTOR - the level of completion of the position, which is updated at every model interval when the position is occupied

Job Records

A line is written to ***ENGR.LIS for each stage of a job. The sequence of data on the line is dependent on the stage, though all job records begin with the following sequence:

"J"

SIDE - the side performing the job ("1" = BLUE, "2" = RED)

MISSION NUMBER - the associated mission number for this job

JOB NUMBER - the job's "serial number" as determined by the entity's memory location (this number and the mission number together form a unique identifier for the job)

CURRENT TIME - the time at which the record was written

STAGE OF THE JOB - a number indicating the current stage of the job:

"1" = create a new job
"2" = activate a new phase
"3" = begin the next segment
"4" = adjust the segment duration
"5" = complete the current segment
"6" = complete the current phase
"7" = complete the job
"8" = complete the mission
"9" = preempt equipment from this job
"10" = discontinue a work team's effort because of attrition
"11" = discontinue the job because of late finish time
"12" = discontinue this job because of cancellation of mission.

For each of the stages of the job, additional information is included:

1 - CREATE A NEW JOB:

TASK TYPE - a number indicating the type of engineer task associated with the job:

..MINEFIELD.BREACH	= 1
..MINEFIELD.CLEAR	= 2
..LINE.OBSTACLE.BREACH	= 3
..BREACH.OBSTACLE.COMPLEX	= 4
..IMPROVE.LINE.BREACH	= 5
..REPAIR.ROAD.CRATER	= 6
..BUILD.TRAIL	= 7
..EMPLACE.MINEFIELD	= 8
..EMPLACE.LINE.OBSTACLE	= 9
..EMPLACE.OBSTACLE.COMPLEX	= 10
..BRIDGE.DEMOLITION	= 11
..PREPARE.POSITION	= 12
..CRATER.ROAD	= 13
..MAINTAIN.ROAD	= 14

FEATURE TYPE - a number indicating the type of feature associated with the job. This varies with the type of task. It is usually the index of the appropriate feature prototype.

TECHNIQUE - the number of the engineer technique originally chosen for this job

RESPONSIBLE HQ UNIT - the index of the engineer HQ assigned to do this job

X,Y LOCATION - the x,y coordinates of the job site

SIZE - the size of the job (nonzero only for certain task types)

START TIME - the earliest clock time at which the job may be started

COMPLETION TIME - the clock time at which the job must be completed

REQUESTING UNIT - the GG unit which requested this job

ORIGINAL MISSION NUMBER - the mission number associated with a job generated by input data (0 if dynamically generated)

OBSTACLE COMPLEX NUMBER - the obstacle complex number from input data if this job concerns part of a complex

OBSTACLE COMPLEX NAME - the name of the obstacle complex prototype if this job concerns part of a complex

2 - ACTIVATE A NEW PHASE:

TECHNIQUE - the number of the engineer technique which is being used for this job

PRIORITY - the number computed for this job during the prioritization procedure which ranks jobs for processing

FIRST SEGMENT NUMBER - the number of the first segment in this phase of the job

LAST SEGMENT NUMBER - the number of the last segment in this phase of the job

3 - BEGIN THE NEXT SEGMENT:

SEGMENT NUMBER - the number of the current segment

SEGMENT DELAY INDICATOR - a number to indicate why the segment duration has been computed to be longer than normal:

"0" = no delay
"1" = delay from lack of full complement of assets
"2" = delay from environmental factors (night, weather, combat)
"3" = delay from a combination of fewer assets and poorer environment

NUMBER OF UNITS CONTRIBUTING EQUIPMENT TO THIS JOB - a count of the number of work teams moving equipment to the task site for the current phase, each one coming from a different base of operations

4 - ADJUST THE SEGMENT DURATION:

SEGMENT NUMBER - the number of the current segment

ADJUSTED TIME - the amount of time in seconds by which the length of the duration has been adjusted to account for losses

5 - COMPLETE THE CURRENT SEGMENT:

SEGMENT NUMBER - the number of the current segment

6 - COMPLETE THE CURRENT PHASE:

SEGMENT NUMBER - the number of the current segment

7 - COMPLETE THE JOB: Superfluous data added during development but not currently used and skipped by the postprocessor

8 - COMPLETE THE MISSION

9 - PREEMPT EQUIPMENT FROM THIS JOB:

SEGMENT NUMBER - the number of the next segment to be processed

10 - DISCONTINUE A WORK TEAM'S EFFORT BECAUSE OF ATTRITION:

SEGMENT NUMBER - the number of the next segment to be processed

11 - DISCONTINUE THE JOB BECAUSE OF LATE FINISH TIME:

SEGMENT NUMBER - the number of the next segment to be processed

DISCONTINUE REASON - a number to indicate why the job is being discontinued:

"0" = the job is still pending and cannot possibly be finished before its latest completion time
"1" = assets already assigned to work on this job are now not sufficient to complete work in time
"2" = environmental factors at the job site will prevent timely completion of the job
"3" = a combination of insufficient assets and poor working conditions will prevent job completion

12 - DISCONTINUE THIS JOB BECAUSE OF CANCELLATION OF MISSION:

SEGMENT NUMBER - the segment number of the last segment completed

COMPLETED EFFECT - the portion of the job completed at this point

Logistics Deficit Records

A line is written when supplies must be transferred between engineer units and the giving unit does not have enough on hand to meet the demand. The simulation continues as if the receiving unit was implicitly supplied, and a record of this type is written to the engineer output.

"L" - the record type

SIDE - the side to which the unit belongs ("1" = BLUE, "2" = RED)

GIVING UNIT - the name of the unit with the supply deficit

CURRENT TIME - the clock time at which the record was written

SUPPLY TYPE REQUESTED - the fuel ammunition, or job supply type that was not available

AMOUNT REQUESTED - the amount of the supply requested

AMOUNT GIVING UNIT HAS ON HAND - the amount of the requested supply actually available to the unit

Unassigned Mission Records

A line is written when it is decided that a mission cannot be assigned, including missions which are still unassigned at the end of the simulation.

"M" - the record type

SIDE - the side for which the mission was generated ("1" = BLUE, "2" = RED)

MISSION NUMBER - the mission's "serial number" as determined by the entity's memory location

CURRENT TIME - the clock time at which the record was written

TASK TYPE - the task type associated with the mission

ORIGINAL MISSION NUMBER - the original mission number from input data if the mission involves obstacle emplacement

X,Y LOCATION - the x,y coordinates of the mission location

START TIME - the earliest clock time at which work on the mission may begin

REQUIRED COMPLETION TIME - the latest clock time acceptable for finishing the mission

NUMBER OF FEATURES - the number of jobs associated with this mission, with work teams assigned to the mission moving from job to job until all features are complete

REQUESTING UNIT - the name of the unit requesting the mission

OBSTACLE COMPLEX - the number of the obstacle complex if this mission is associated with emplacing or breaching a part of a complex

OBSTACLE NAME - the name of the obstacle complex if this mission is associated with emplacing or breaching a part of a complex

Depending on the task, other lines are written giving more information about the specific features associated with the mission.

MINEFIELD BREACH:

If the mission involves a single, stand-alone minefield, the following information is written:

MINE TYPE - the name of the minefield prototype

FRONTAGE - the frontage of the minefield in kilometers

DEPTH - the depth of the minefield in kilometers

NUMBER OF MINES - the number of mines remaining in the minefield

ORIENTATION - the minefield's orientation

X,Y LOCATION OF THE MINEFIELD - the x,y coordinates of the center of the minefield

If the mission involves breaching a part of an obstacle complex, several minefields may be included. For each minefield, the following information is written:

MINE TYPE - the name of the minefield prototype

FRONTAGE - the frontage of the minefield in kilometers

DEPTH - the depth of the minefield in kilometers

NUMBER OF MINES - the number of mines in the minefield

COMPLETION PERCENTAGE - the level of completion associated with this obstacle type

BREACHER MISSION NUMBER - the mission number associated with this feature type in the MF input data

MINEFIELD CLEAR:

MINE TYPE - the name of the minefield prototype

FRONTAGE - the frontage of the minefield in kilometers

DEPTH - the depth of the minefield in kilometers

NUMBER OF MINES - the number of mines remaining in the minefield

ORIENTATION - the minefield's orientation

X,Y LOCATION OF THE MINEFIELD - the x,y coordinates of the center of the minefield

LINE OBSTACLE BREACH:

If the line obstacle is not a part of an obstacle complex, the following information is written:

OBSTACLE TYPE - the line obstacle prototype of tne obstacle

X,Y LOCATION OF THE START OF THE SEGMENT - the location of the start of the segment to be breached

X,Y LOCATION OF THE END OF THE SEGMENT - the location of the end of the segment to be breached

X,Y LOCATION OF THE BREACH OF THE SEGMENT - the breach location

If the line obstacle is part of an obstacle complex, the following information is written:

LINE OBSTACLE TYPE - the line obstacle prototype of the obstacle

FRONTAGE - the length of the line obstacle

DEPTH - normally 0, written to give consistency with minefield feature reports

COMPLETION PERCENTAGE - the level of completion associated with this obstacle type in the complex

BREACHER MISSION NUMBER - the number of the breacher mission associated with this feature in the MF input data for this complex prototype

BREACH OBSTACLE COMPLEX:

OBSTACLE COMPLEX NAME - the name of the obstacle complex prototype of the complex being breached as a single mission

IMPROVE LINE BREACH:

LINE FEATURE TYPE - the prototype of the line feature containing the breach

X,Y LOCATION OF THE START OF THE SEGMENT - the location of the start of the line feature containing the breach

X,Y LOCATION OF THE END OF THE SEGMENT - the location of the end of the line feature containing the breach

X,Y LOCATION OF THE BREACH OF THE SEGMENT - the location of the breach site

EMPLACE MINEFIELD:

If the mission involves minefields which are not part of a complex, a line of information is written for each minefi^ld in the mission in the following format:

MINF TYPF - the minefield prototype

FRONTAGE - the minefield's frontage in kilometers

DEPTH - the minefield's depth in kilometers

NUMBER OF MINES - the number of active mines in the minefield

ORIENTATION - the minefield's orientation

X,Y LOCATION OF THE MINEFIELD - the location of the minefield's center

If the mission involves minefields that are part of an obstacle complex, the following information is written:

MINE TYPE - the minefield prototype

FRONTAGE - the minefield's frontage in kilometers

DEPTH - the minefield's depth in kilometers

NUMBER OF MINES - the number of active mines in the minefield

COMPLETION PERCENTAGE - the level of completion of this obstacle type

EMPLACER MISSION NUMBER - the mission number associated with emplacing this type of obstacle in the obstacle complex prototype data

EMPLACE LINE OBSTACLE:

If the line obstacle is not part of an obstacle complex, a line of information is written for each obstacle in the mission as follows:

LINE OBSTACLE TYPE - the obstacle prototype

X,Y LOCATION OF THE START OF THE OBSTACLE - the location of the start of the obstacle

X,Y LOCATION OF THE END OF THE OBSTACLE - the location of the end of the obstacle

FEATURE NUMBER - the feature number of this obstacle from the TB input data

If the line obstacle is part of an obstacle complex, the following information is written:

LINE OBSTACLE TYPE - the obstacle prototype

FRONTAGE - the length of the obstacle in kilometers

DEPTH - normally 0, written for consistency with minefield obstacle types

COMPLETION PERCENTAGE - the level of completion for this obstacle type

EMPLACER MISSION NUMBER - the number of the emplacement mission associated with this obstacle in the complex prototype data

EMPLACE OBSTACLE COMPLEX:

OBSTACLE COMPLEX NAME - the name of the obstacle complex prototype being emplaced as a single mission

PREPARE POSITION:

DEFENSIVE POSITION TYPE - the name of the position prototype

X,Y LOCATION - the x,y coordinates of the position location

UNIT NAME - the name of requesting or occupying unit

COMPLETION FACTOR - the level of completion of the position

Implicit Engineer Job Records

Nonengineer units are capable of performing engineer tasks if they own the appropriate resources and are identified as the requesting unit. A line in the following format is written upon completion of such a job.

"N" - the record type

SIDE - the side performing the work ("1" = BLUE, "2" = RED)

UNIT - the name of the unit performing the work

START TIME - the time at which the work began

TASK TYPE - the number associated with the type of task being performed

FEATURE TYPE - the index of the feature prototype for the feature being altered by this work, which varies according to the task type

TECHNIQUE - the engineer technique being used to perform the work

X,Y LOCATION - the x,y coordinates of the work site

DURATION - the length of the work time in hours

COMPLETED EFFECT - the portion of the job that the unit was able to complete

The File ***ENSTP.LIS

At the end of the simulation run, a final report is sent to the file ***ENGR.LIS for each unassigned mission. These records are in the same format used earlier for rejected missions.

Another file, ***ENSTP.LIS, receives information needed to create the appropriate data structures used by the engineer postprocessor:

THE TIME THAT THE SIMULATION ENDED

THE TOTAL NUMBER OF MISSIONS CREATED

THE NUMBER OF ENGINEER ASSET PROTOTYPES

THE FL WEAPON NAME FOR EACH ASSET PROTOTYPE

THE NUMBER OF BLUE HQ UNITS

THE NUMBER OF RED HQ UNITS

THE GG NAME OF EACH BLUE UNIT

THE GG NAME OF EACH RED UNIT

The last two data items indicate the number of BLUE and RED engineer teams that were not needed during the run. In future runs of the same scenario, the numbers of BLUE and RED work teams entered in the GG data module may be reduced by the amounts printed here so as to optimize the number of extra units.

The Engineer Postprocessor

The engineer postprocessor, ENPOST, uses ***ENSTP.LIS and ***ENGR.LIS as its input files. It reads through the records in ***ENSTP.LIS to establish its data structures and then reads ***ENGR.LIS to accumulate the data.

To run ENPOST, the current directory must contain the files ***ENSTP.LIS and ***ENGR.LIS. Also, the commands must be issued to the operating system to assign ***ENGR.LIS to SIMU21 and ***ENSTP.LIS to SIMU58. The *** must be replaced by the prefix used for the output to be processed. Note that such commands already will have been issued if VIC was run immediately before using the postprocessor.

The postprocessor prepares several reports, both detail and summary, for jobs and assets. It also prepares detailed reports for unassigned missions, logistics deficits, bridging activity, defensive position usage and nonengineer job performance.

The reader is advised to consult the separate technical report for the postprocessor for more information regarding standard reports and the structure of database files produced for later processing by database management systems.

5 ENGINEER DATA STRUCTURES

Introduction

The data structures of the EN module are described below. Included here are descriptions of each entity and its attributes and the sets that establish relationships between entities. The order of presentation follows the logical flow of the relationships rather than the formal structure of the preamble.

The units of measure associated with each attribute are the units used internally. Attributes with values determined by input data may have internal units of measure different from the units required in the input data. For example, all times are converted to seconds internally, but input data may have specified a preparation time in minutes and a rest time in hours.

Variables, Attributes, and Sets Added to Non-EN Entities

EN.TIMING.ARRAY (SYSTEM ARRAY) - a 2 by 4 real array containing timing information for engineer functions. Input data specifies a mission processing interval, a planning interval, a job turnaround interval, and a task reorganization interval for each side. Except for the planning interval, these times must be nonzero since they are used for cyclic event scheduling. A zero planning time will prevent planning for mobility tasks.

The mission processing interval is the maximum length of time allowed between successive passes through the side's mission list in an attempt to assign the missions. Each side maintains a list of its unassigned engineer missions. When the list is processed, all missions that can be assigned are removed from the mission list and converted to engineer jobs. If this processing interval is short and the mission list contains a large number of missions which cannot be assigned because of current conditions, the run time may be affected. On the other hand, the larger this interval is, the less responsive engineers will be to requests for support. Dynamically generated mobility missions are processed immediately and do not use this interval.

The planning interval is the maximum amount of time allowed between a unit's recognition that it must plan for a known obstacle in its path and its actual encounter with that obstacle. Each unit scans its path for known obstacles and creates a planning path point for each. This path point is positioned in the path to allow as much time as possible, up to a maximum of the planning interval, for the unit to request engineer assistance. A path point arrival at the planning point generates an engineer mission to breach the obstacle.

The event EN.PROCESS.JOB.LIST, the mechanism for initiating work on pending jobs, is scheduled for a staff unit when work teams return to their base, new supplies are delivered, new jobs are assigned, and a unit's path direction changes. The job turnaround interval is used to schedule this event in the first three situations, and the new role interval is used to schedule this event in the last situation. Neither of these intervals may be zero, as this would produce an infinite loop in the timing routine. The job turnaround interval represents the time required to process equipment and supplies in and out, given that planning and scheduling have already been accomplished. In the case of a new job being assigned to the staff, the assumption is that the staff was adequately forewarned or that the task is one that is normally handled spontaneously. Note that each type of equipment already accounts for a preparation time before leaving its base with a new

assignment, so that time should not be considered in the job turnaround time. The new role interval represents the time required by staff to assess a changed situation and to plan the supporting engineer activities, including reprioritizing its active and pending jobs and perhaps canceling jobs already in progress.

EN.MISSION.COUNT (SYSTEM VARIABLE) - an integer used to attach a unique identifier to engineer missions as they are assigned to an engineer HQ unit, linking all of the jobs arising from the mission by a common number. As a mission is processed, each job in the mission has the current value of this number as its mission number. After a mission has been numbered and assigned, EN.MISSION.COUNT is incremented by 1.

EN.TECHNIQUE.ARRAY (SYSTEM ARRAY) - a ragged integer array, dimensioned by side by task type by obstacle feature prototype by the number of corresponding techniques available, providing an index to the engineer techniques. It is built immediately after the technique data is read.

EN.PRIORITY.ARRAY (SYSTEM ARRAY) - an integer array, dimensioned by side by combat status mapping by engineer mission type, coming from the engineer input data and providing the first step in establishing priorities for the jobs on a staff unit's job list. The GG input data associates a combat status mapping (ATTACK, DEFEND, WITHDRAW) with each combat status; internally, each engineer task type has a mapping into one of the four engineer mission types (MOBILITY, COUNTERMOBILITY, SURVIVABILITY, GENERAL ENGINEERING). This array links each combination of combat status mapping and engineer mission type with a relative priority level.

EN.SET.OF.FREE.WPN.GROUPS (SYSTEM OWNED SET) - a set containing the FL.WPN.GROUPs not currently owned by a GG.UNIT. Since FL.WPN.GROUPs are permanent entities, extra weapon group must be created at initialization time for use when work teams are created and must have weapon groups to hold their required equipment.

SS.SIDE (PERMANENT ENTITY) - a permanent entity created in the SS module for each side in the simulation, having attributes related to most of the other modules of VIC. The EN module added several attributes and sets.

EN.MAX.FREE.WORK.TEAMS (ATTRIBUTE) - a variable to keep track of the smallest number of units left in a side's pool of free work teams during a run. It is printed in the engineer set up file ***ENSTP.LIS at the end of a run. Each side has a pool of unused units from which to form engineer work teams as they are required. Determining the correct initial size of these pools is a difficult task. With the value of this variable known for each side, successive runs of the same scenario may decrease the number of free units by this amount so as to optimize the number of extra units created and thereby decrease run time.

EN.SET.OF.FREE.UNITS (SET OWNER) - a set to keep track of free work units. An engineer work team is either in this set or in the performing units set of the HQ unit responsible for the jobs assigned to the work team.

EN.SET.OF.HQ.UNITS (SET OWNER) - a set to keep track of the engineer HQ units on a side.

EN.SET.OF.STAFF.UNITS (SET OWNER) - a set to keep track of the engineer staff units on a side.

EN.SET.OF.MISSIONS (SET OWNER) - a set to hold the information about unassigned engineer activity for the side. This set grows and shrinks as engineer activities are generated and assigned. It is processed at regularly scheduled intervals, as determined by the mission update interval in the engineer timing array. Each mission which can be assigned is removed from the set, and corresponding jobs are created and filed in the responsible HQ's set of pending jobs. Those missions which cannot be assigned remain in this set awaiting a change in conditions.

EN.SET.OF.PROPOSED.MINEFIELDS (SET OWNER) - a set containing all of the MF.MINEFIELDs which are not active and which are to be emplaced by engineers on this side.

EN.SET.OF.PROPOSED.OBSTACLES (SET OWNER) - a set containing all of the TB.OBSTACLE. SECTIONs which are not active and which are to be emplaced by engineers on this side.

EN.SET.OF.PROPOSED.COMPLEXES (SET OWNER) - a set containing all of the MF.OBSTACLE. COMPLEXs which are not active and which are to be emplaced by engineers on this side.

EN.SET.OF.DEFENSIVE.POSITION (SET OWNER) - a set containing all of the defensive positions presently in place and those to be emplaced by engineers on this side, with the EN.DP.COMPLETION. FACTOR ranging from ..UNPREPARED (0.0) for future positions to ..FULLY.PREPARED (1.0) for active positions.

EN.SET.OF.RETRIEVABLE.ASSETS (SET OWNER) - a set to keep track of retrievable assets belonging to the side. EN.REQUIRED.EQUIPMENT with EN.EQ.USE set to ..RETRIEVABLE does not return from the job site. In the case of bridging assets, assets left at the site may be retrieved later for use on another job or may be destroyed by enemy units controlling the bridge site.

Engineer Task Structures

EN.DEF.POS.PROTO (PERMANENT ENTITY) - a type of defensive position, as determined by the unit type and level for units wishing to use a position of this type. Formerly, defensive positions were represented as a unit path point attribute. This new structure allows the representation of different types of positions with varying levels of engineer and maneuver unit effort to emplace them and varying levels of protection from indirect fire. Additionally, this structure controls the use of each position by units other than the unit for which it was originally prepared and the destruction of positions when they fall into enemy hands.

EN.DP.PR.NAME (ATTRIBUTE) - the defensive position prototype name to be used for input and output.

EN.DP.PR.UNIT.TYPE (ATTRIBUTE) - the type of unit for which this position is prepared. Its possible values are in the set of defined-to-means for GG.TYPE as well as the value zero.

EN.DP.PR.UNIT.LEVEL (ATTRIBUTE) - the level of unit for which this position is prepared. Its possible values are in the set of defined-to-means for GG.LEVEL as well as the value zero.

EN.DEFENSIVE.POSITION (TEMPORARY ENTITY) - an actual occurrence of a defensive position on the battlefield. This structure serves the same purpose for defensive positions as the TB.LINE. FEATURE.SEGMENT for linear obstacles and the MF.MINEFIELD for minefields.

EN.DP.PROTO (ATTRIBUTE) - the prototype number of the EN.DEF.POS.PROTO for this position.

EN.DP.SIDE (ATTRIBUTE) - the emplacing side for the position. A unit looking for a position within its deployment radius will destroy any position it finds which does not have this attribute set to its own side.

EN.DP.X.LOCATION (ATTRIBUTE) - the x-coordinate of the defensive position's location.

EN.DP.Y.LOCATION (ATTRIBUTE) - the y-coordinate of the defensive position's location.

EN.DP.UNIT (ATTRIBUTE) - the unit for which the position was originally prepared. If engineers have moved ahead of the unit and completed the position before the unit arrived or if the position was declared fully prepared at the start of the simulation, no other unit may occupy this position until the current unit has come and gone.

EN.DP.COMPLETION.FACTOR (ATTRIBUTE) - the percentage of completion of the position. It becomes the GG.DEF.POS.STATUS of the occupying unit.

EN.DP.EXPOSURE.FACTOR (ATTRIBUTE) - an interpolation factor used to determine the indirect fire exposure of the occupying unit. The combat status data from the GG input gives a portion of unit exposed to indirect fire in both a prepared and an unprepared position. For positions prepared using engineer technique data, the relative effect of the technique used to prepare the position is recorded as the exposure factor for the position. It combines with the level of completion of the position for the linear interpolation between the two levels of exposure. This allows different engineer techniques to produce the same level of protection from direct fire but different levels of protection from indirect fire. When the relative effect of the technique, and therefore the position's exposure factor, is zero, the position offers no added protection from indirect fire. When it is one, the position gives maximum protection from indirect fire.

EN.SET.OF.DEFENSIVE.POSITIONS (SET MEMBER) - the set of defensive positions which may be used by units of the side as they move across the battlefield.

EN.MISSION (TEMPORARY ENTITY) - a structure for holding the data for a desired engineer activity until an engineer HQ unit can be found to do the work. Once the task is assigned to an HQ unit, the EN.JOB structure is created, and the mission is removed from its side's set of missions and destroyed.

EN.MISSION.TASK (ATTRIBUTE) - the engineer task type associated with the mission, using defined-to-means.

EN.MISSION.ORIG.NBR (ATTRIBUTE) - the original mission number assigned in the input data, recorded here in order to keep track of missions specified in the GG, MF, and TB input data. Using the task type and original mission number together will allow the user to track specific missions generated by input data. This number is not related to the actual mission number associated with the jobs which arise from it. See EN.MISSION.COUNT.

EN.MISSION.METHOD (ATTRIBUTE) - a defined-to-mean which specifies how to search for a technique for accomplishing this mission. In choosing a technique, the search may be directed to follow one of three paths: ..PREFERRED.METHOD (the default), which takes the first usable technique listed in the input data; ..SHORTEST.TIME, which takes the technique which requires the shortest amount of time to complete; ..BEST.EFFECT, which takes the technique which yields the best end result. The situation which generated the mission will determine the method.

EN.MISSION.X.LOC (ATTRIBUTE) - the x-coordinate of the mission location. For some of the tasks, this is redundant information while for others it is necessary. For example, for emplacing a set of minefields, this location will be ignored in favor of using the centers of the minefields as determined by the feature below. For breaching of linear obstacles, however, this location will identify the actual breach location, while the feature information only specifies the endpoints of the segment.

EN.MISSION.Y.LOC (ATTRIBUTE) - the y-coordinate of the mission location. See EN.MISSION.X.LOC above.

EN.MISSION.REQUESTOR (ATTRIBUTE) - the GG.UNIT for which this mission was generated. For obstacle emplacement missions created by input data, the requestor is an optional data item. Each dynamically generated mission will be done for the specific unit which generated it. That unit's GG.UNIT number is stored here and is used in searching for engineer support. An attempt will be made to assign the engineer mission to an engineer unit subordinate to the requestor.

EN.MISSION.CREATION.TIME (ATTRIBUTE) - the clock time at which the mission was generated.

EN.MISSION.REQUIRED.COMPLETION.TIME (ATTRIBUTE) - the latest acceptable completion time for the mission. For obstacle emplacement missions generated by input data, the required completion time is generally the input start time for the obstacle. For dynamically generated jobs, an analysis of the situation which generates the job will produce a time at which the work must be completed. This is an important decision variable, with a job being canceled if its estimated completion time exceeds this value.

EN.MISSION.START.TIME (ATTRIBUTE) - the earliest time at which engineers may begin work on this mission. Dynamically generated missions can begin as soon as they are generated. Obstacle emplacement missions, however, need this extra control mechanism as a part of the input data to help organize the engineer effort.

EN.MISSION.OC.PTR (ATTRIBUTE) - the obstacle complex that this mission involves. An obstacle complex mission has the possibility of dividing into its component sub-missions, with the sub-missions having different tasks and feature from the mission. This attribute allows such sub-missions to keep track of the parent obstacle complex. This pointer is zero except for subdivided obstacle complex missions.

EN.SET.OF.POSSIBLE.UNITS (SET OWNER) - a set to keep track of the units which may possibly be assigned this mission. In order to minimize the processing required to determine which unit should be assigned a mission, this set keeps track of all units found during the first assignment cycle that are capable of doing the work. In successive cycles, only these units are

checked to determine if any have now moved to a position that will allow the assignment to take place.

EN.SET.OF.MISSION.FEATURES (SET OWNER) - the set of features associated with the mission. One mission will generate a number of engineer jobs, one for each terrain feature being worked on by the work team created for this mission. Each job corresponds to a particular feature. For example, if the task is to emplace minefields, this set is the set of pointers to the proposed minefields which are to be constructed by one work team.

EN.SET.OF.MISSIONS (SET MEMBER) - the set containing the records for all engineer activity which has been generated but not yet assigned to specific engineer HQ units. Each side owns a set of engineer missions.

EN.MISSION.UNIT.POINTER (TEMPORARY ENTITY) - an entity to keep track of a unit and its technique that have been determined as fitting this mission's requirements for assignment.

EN.MISSION.UNIT (ATTRIBUTE) - the GG.UNIT number of the unit which is a possibility for being assigned this mission.

EN.MISSION.TECHNIQUE (ATTRIBUTE) - the technique which the given unit has determined it can use to accomplish the mission. This is the first technique for the given side, task, and feature for which the unit has the equipment and supplies. This need not be the technique that this unit ultimately uses if the mission is assigned to it; it will be the first technique in the list that the unit checks.

EN.SET.OF.POSSIBLE.UNITS (SET MEMBER) - a set to keep track of the units that may be assigned this mission. In order to minimize the processing required to determine which unit should be assigned a mission, this set keeps track of all units found during the first assignment cycle that are capable of doing the work. In successive cycles, only these units are checked to determine if any have now moved to a position that will allow the assignment to take place.

EN.MISSION.FEATURE (TEMPORARY ENTITY) - a entity created to keep track of the terrain feature being changed by the accomplishment of this mission.

EN.MISSION.FEATURE.PTR (ATTRIBUTE) - a pointer to the terrain feature (the proposed MF.MINEFIELD, the proposed TB.OBSTACLE, the TB.LINE.FEATURE.SEGMENT which is to be breached, the EN.DEFENSIVE.POSITION which is to be constructed, etc.)

EN.SET.OF.MISSION.FEATURES (SET MEMBER) - the set of all features associated with the mission. This set provides the mechanism for linking jobs in one mission to be carried out by one work team.

Engineer Units

EN.HQ.UNIT (TEMPORARY ENTITY) - the basic engineer unit in the simulation. These units appear in the unit database of the input data; they own the engineer assets and are responsible for engineer jobs. To do a job, the HQ unit creates a work team and transfers to it the equipment and supplies required by the job's chosen technique.

EN.HQ.UNIT.ID (ATTRIBUTE) - the index of the GG.UNIT corresponding to this HQ unit.

EN.HQ.INDEX (ATTRIBUTE) - an index added to ease processing the engineer history file in the postprocessor. The engineer HQ units are numbered consecutively from 1 through the sets of blue and red HQ units. VIC engineer output uses these numbers for reference and associates each engineer HQ unit with a permanent entity with this index in the postprocessor.

EN.WG.STRENGTH.PTR (ATTRIBUTE) - the address of a 2 by N.EN.ASSET.PROTO double array used by the HQ to keep track of the strength of its engineer assets that are available and at work.

EN.HQ.ASSET.PTR (ATTRIBUTE) - the address of a 2 by N.EN.ASSET.PROTO integer array used by the HQ to keep track of the number of pieces of each type of equipment it has available and the address of the first piece of each type of equipment in its inventory.

EN.HQ.INVENTORY (SET OWNER) - the set of records used to track the engineer assets of a type for which the input report flag is on. The HQ unit maintains an inventory of all of such engineer assets. Except for the ordering of its records, an HQ's inventory changes only when assets are transferred to or from another HQ unit in a permanent transfer. Otherwise, each piece of equipment in the inventory has a record of which GG.UNIT currently has it, its current job, its current level of damage, etc.

EN.SET.OF.HQ.PENDING.JOBS (SET OWNER) - the set of jobs which have been assigned to an HQ unit but which are not presently active. When engineer jobs arise, they are assigned to an HQ unit according to a fixed set of rules, prioritized according to the situation, and placed on this list.

EN.SET.OF.HQ.ACTIVE.JOBS (SET OWNER) - the set of jobs which have been assigned to an HQ unit and have work teams assigned to work on them. When the HQ's staff is ready to process a pending job and determines that all required equipment for it is available for assignment, work teams are created to bring the equipment together to do the job, and the job is moved to the active list. When a phase is completed, the job may move back to the pending list to wait for assets for the next phase.

EN.SET.OF.PERFORMING.UNITS (SET OWNER) - the set of all work teams which have been created to work on jobs for which this HQ is responsible. Engineer work teams are either in the free pool (not working) or on this list of the HQ unit which is responsible for the job that the work team is performing.

EN.SET.OF.HQ.UNITS (SET MEMBER) - the set that keeps track of the HQ units. Each side in the simulation has such a set.

EN.STAFF.UNIT (TEMPORARY ENTITY) - the decision-making engineer unit. A GG.UNIT of type ..ENGINEER whose superior is not of type ..ENGINEER is an engineer staff unit. The staff unit processes a consolidated job list to determine which pending jobs of the units in its command chain should be started, with equipment owned by such units being shared by all other units in the chain.

EN.STAFF.UNIT.ID (ATTRIBUTE) - the index of the corresponding GG.UNIT.

EN.STAFF.PROCESS.EVENT (ATTRIBUTE) - a pointer to the EN.PROCESS.JOB.LIST event which is the next event for processing this staff's job list. Elements of the engineer timing array are used to schedule and reschedule this event according to the occurrence of certain situations.

EN.STAFF.ASSET.PTR (ATTRIBUTE) - the address of a 1 by N.EN.ASSET.PROTO array whose nth element is the total amount of equipment of type n owned by the command chain headed by this staff engineer. The amount changes only if equipment is damaged or formally transferred into or out of this command.

EN.STAFF.TECH.PTR (ATTRIBUTE) - the address of a 2-dimensional array indexed by task type and feature type, whose entries are the numbers of the first corresponding techniques for which the required equipment is available to this staff unit. If the entry in element m,n is zero, this staff unit is not capable of performing a job of task type m, feature type n. Otherwise, the entry is the technique number of the first technique in the input data that is available to this command for this task/feature combination.

EN.STAFF.RANKING.LEVEL (ATTRIBUTE) - the level of the staff unit's GG.UNIT prototype. It is used to rank the units in the set of staff units, with units of low rank being filed first.

EN.SET.OF.STAFF.JOBS (SET OWNER) - a set of job pointers, one for each of the jobs on the active and pending lists of each of the HQ units in this staff unit's command chain. This list is rebuilt when reprioritization of jobs is necessary. It is used as a working set, being constructed to put all current jobs in the command chain in one place for prioritization and then to process the pending jobs in priority order.

EN.SET.OF.STAFF.UNITS (SET MEMBER) - the set that keeps track of the staff units, which are temporary entities. Each side has such a set.

EN.WT.UNIT (TEMPORARY ENTITY) - the engineer GG.UNIT used to simulate the work process. Engineer work teams are created with the rest of the GG.UNITs but do not have their attributes set by input data. These units are treated as spare GG.UNITs when they are not assigned to a specific job. When they are assigned to a job, they are in the set of performing units for the HQ unit which is responsible for the job.

EN.WT.UNIT.ID (ATTRIBUTE) - the index of the GG.UNIT associated with this work team.

EN.WT.RESPONSIBLE.HQ (ATTRIBUTE) - the HQ unit which is responsible for the jobs assigned to the work team when it is active; 0 otherwise.

EN.WT.ASSET.PTR (ATTRIBUTE) - a pointer to a 1 by N.EN.ASSET.PROTO array whose nth entry is the number of assets of type n held by the work team.

EN.WT.JOB.PRIORITY (ATTRIBUTE) - the priority of the first job on the work team's list or 0.0 if the work team is returning from a job. When a search is being made for equipment to preempt, this is the attribute that is checked to determine whether this work team's equipment may be preempted.

EN.WT.MAX.SPEED (ATTRIBUTE) - the maximum speed the active work team can travel, based on the equipment held by the work team. It is computed after the work team is formed and its path is determined. It is used to retard the ground unit prototype speed of an engineer work team if the work team owns equipment which cannot move at the prototype speed.

EN.SET.OF.WT.ASSETS (SET OWNER) - the work team's record-keeping device for the equipment that has been temporarily transferred to it by an HQ unit. Since each engineer asset is being tracked throughout the simulation, when a piece of equipment is transferred between an HQ unit and a work team or between two work teams, a record must be kept showing who owns the equipment, which piece of equipment it is associated with in the owning HQ's inventory, etc.

EN.SET.OF.WT.JOBS (SET OWNER) - the set of jobs which the work team has been assigned to do sequentially. A work team is activated to work on a specific phase of a set of jobs in one mission for which a particular HQ unit is responsible. For each of those jobs, the main information about the job is kept with the HQ unit as an EN.JOB, but a record of the crucial information about the job for the work team is kept in EN.WT.JOB. Each EN.WT.JOB must have a corresponding EN.JOB, while many EN.WT.JOBs may correspond to only one EN.JOB.

EN.SET.OF.FREE.UNITS (SET MEMBER)
EN.SET.OF.PERFORMING.UNITS (SET MEMBER) - the two types of sets which keep track of the work teams. Work teams must be in one of two places. If not active, they are filed in the free pool for their side. When working on a job (or set of jobs), they are filed in the performing units set of the HQ unit responsible for the job in the order of EN.WT.JOB.PRIORITY, with lowest priority teams at the top of the list to aid the search for assets.

Engineer Assets

EN.ASSET.PROTO (PERMANENT ENTITY) - a type of engineer equipment. For each type of engineer equipment being modeled, an asset prototype is created. Its attributes hold information about that equipment type.

EN.ASSET.PR.REPORT (ATTRIBUTE) - a flag which indicates whether or not detailed records are to be kept for assets of this type. If the flag is off, no inventory records are created for assets of this type, no asset output reports are written to the engineer history file, no attempt is made to track required rest periods for this type of asset, and attrition of this type of asset is not tracked at the individual item level.

EN.ASSET.PR.WP.PTR (ATTRIBUTE) - the index of the weapon prototype corresponding to this engineer asset type. Input data specifies equipment by name in both the FL and the EN modules. This attribute is set by matching the names from the FL weapon prototype list with the names in the EN input data.

EN.ASSET.PR.SPEED (ATTRIBUTE) - the ground speed of this equipment under good trafficability and good weather. This attribute must be nonzero since engineer work teams cannot use platforms and set their speed as the minimum value of this attribute for the equipment types owned.

EN.ASSET.DAMAGE.THRESHOLD (ATTRIBUTE) - the minimum usable portion which must be functioning for a piece of this type of equipment to be active. This information is necessary for modeling attrition of engineer assets. An asset which has damage at a level above this threshold suffers a degradation of work capacity which is reflected in the calculation of its job's duration. Below this threshold, a piece of equipment is declared ..DAMAGED and is no longer available for work.

EN.ASSET.PR.MAX.WORK.PERIOD (ATTRIBUTE) - the maximum length of time in seconds that this type of equipment can work before requiring a rest.

EN.ASSET.PR.REST.PERIOD (ATTRIBUTE) - the length of time in seconds that this type of equipment must rest after a maximum work period before it can work again.

EN.ASSET (TEMPORARY ENTITY) - an actual piece of engineer equipment. It is owned by an engineer HQ unit and kept in that HQ unit's inventory.

EN.ASSET.PROTO.ID (ATTRIBUTE) - the index of this asset's prototype.

EN.ASSET.UNIT.ASSIGNMENT (ATTRIBUTE) - the index of the GG.UNIT which has physical possession of the asset.

EN.ASSET.STATUS (ATTRIBUTE) - the current status of the asset, used to track the activity of equipment, corresponding to defined-to-means:

..AVAILABLE	TO MEAN 1
..PRE.TASK	TO MEAN 2
..IN.TRANSIT	TO MEAN 3
..WAITING	TO MEAN 4
..WORKING	TO MEAN 5
..IN.REST.PERIOD	TO MEAN 6
..DAMAGED	TO MEAN 7
..LEFT.AT.SITE	TO MEAN 8

EN.ASSET.UNDAMAGED.PORTION (ATTRIBUTE) - the current usable portion of the asset. When attrition is assessed, this number decreases from 1.0 (undamaged) until the level of damage is below the damage threshold for this type of asset. Between those two levels, the performance capability of this asset decreases linearly.

EN.ASSET.LAST.UPDATE (ATTRIBUTE) - the clock time at which the last update of attributes was done. Used to measure hours worked, hours at rest, etc.

EN.ASSET.HOURS.WORK.SINCE.REST (ATTRIBUTE) - a cumulative total of time worked since rest, actually kept in seconds instead of hours.

EN.ASSET.CURRENT.JOB (ATTRIBUTE) - the pointer to the EN.JOB for which this asset is currently being used. An asset can be working on only one job at a time.

EN.HQ.INVENTORY (SET MEMBER) - the set used to keep track of assets owned by an HQ. The EN.ASSET and its attributes serve as an inventory record in the HQ unit's EN.HQ.INVENTORY.

EN.RETRIEVABLE.ASSET (TEMPORARY ENTITY) - an entity created to keep track of engineer assets left at a work site. When a technique has required equipment marked with an EN.EQ.USE = ..RETRIEVABLE, that equipment is removed from the work team at the job site and made a retrievable asset. This is a record of what type of asset it is, where it is, and who owns it.

EN.REA.PROTO (ATTRIBUTE) - the asset prototype index of the equipment.

EN.REA.FEATURE (ATTRIBUTE) - the actual linear obstacle, minefield, etc. containing the equipment.

EN.REA.X.LOCATION (ATTRIBUTE) - the x-coordinate of the asset's location.

EN.REA.Y.LOCATION (ATTRIBUTE) - the y-coordinate of the asset's location.

EN.REA.OWNER (ATTRIBUTE) - the HQ unit owning the equipment.

EN.REA.INV.ID (ATTRIBUTE) - a pointer to the actual EN.ASSET in the EN.HQ.INVENTORY. Note that only equipment whose type has a report flag on may be marked as retrievable.

EN.SET.OF.RETRIEVABLE.ASSETS (SET MEMBER) - a list of retrievable assets belonging to a side.

EN.WT.ASSET (TEMPORARY ENTITY) - a work team record for an engineer asset it temporarily possesses. When an HQ unit transfers an asset to a work team, the work team must create a record for that item so that all of the information necessary for using and returning the asset can be stored.

EN.WT.ASSET.PROTO (ATTRIBUTE) - the index of the asset prototype for this asset.

EN.WT ASSET.OWNER (ATTRIBUTE) - the HQ unit which owns the asset.

EN.WT.ASSET.INV.ID (ATTRIBUTE) - the pointer to the EN.ASSET in the HQ's inventory corresponding to this asset if this is a type of asset for which such records are created, i.e., if the report flag is on.

EN.WT.ASSET.PORTION (ATTRIBUTE) - the undamaged portion of this asset if it is not the type of asset for which inventory records are created. This attributes tracks damage to assets which have no inventory records so that attrition may be properly distributed. An asset of this type is assumed to be at 100 percent effectiveness when it leaves the HQ unit. Any damage to the asset while it is at work is recorded here.

EN.SET.OF.WT.ASSETS (SET MEMBER) - a list of work team assets which serves as the work team's inventory.

Engineer Techniques

EN.TECHNIQUE (PERMANENT ENTITY) - the entity whose attributes specify a particular way in which to perform an engineer task, as determined by side, task type, and feature type.

EN.TECH.SIDE (ATTRIBUTE) - the side (..RED or ..BLUE) that uses this technique.

EN.TECH.TASK (ATTRIBUTE) - the task type (a define-to-mean) for which this technique is used.

EN.TECH.FEATURE.PROTO (ATTRIBUTE) - the feature prototype on which this technique is used.

EN.TECH.COMMAND.LEVEL (ATTRIBUTE) - the GG.PROTOTYPE unit level (a defined-to-mean) of the type of unit which usually employs this technique.

EN.TECH.MIN.DISTANCE.FROM.FEBA (ATTRIBUTE) - the minimum signed distance in kilometers from the FEBA under which this technique may not be used.

EN.TECH.ORGANIZATION.PT (ATTRIBUTE) - a flag with defined-to-mean values which allows data input to dictate whether units coming from a number of locations to work on a specific job will rendezvous at the responsible HQ unit's location or at the work site. Its possible values are:

..AT.HQ	TO MEAN 0
..AT.SITE	TO MEAN 1

EN.TECH.EFFECT (ATTRIBUTE) - an attribute which allows the technique to have a different end effect from other techniques for the same task and feature type. Only five tasks use this relative effect. For linear obstacle breach and improve linear obstacle breach, the standard delay associated with the breached obstacle is divided by this relative effect (any positive real number) for breaches constructed with this technique. For minefield breaches, the relative effect ranges from 0.0 to 1.0 and is a multiplier of the minefield clearance factor. For position preparation, the relative effect is an additional interpolation factor for unit exposure to indirect fire, ranging from no protection (0.0) to maximum protection (1.0) from indirect fire. For building combat trails, the relative effect is rounded to an integer to become the number of levels of improvement in trafficability afforded by the trail.

EN.TECH.ASSET.PTR (ATTRIBUTE) - a pointer to a 1 by N.EN.ASSET.PROTO array whose nth element is the minimum number of assets of type n required for this technique. This array is derived from the required equipment data to provide a quick decision tool for comparing equipment required with equipment available.

EN.SET.OF.TECH.SEGMENTS (SET OWNER) - the set which keeps track of a technique's segments. The technique breaks the job into pieces called segments, each with its own duration, end effect, and timing factors.

EN.SET.OF.REQ.EQUIPMENT (SET OWNER) - a list of the equipment required for this technique.

EN.SET.OF.REQ.SUPPLIES (SET OWNER) - a list of the supplies required for this technique. A distinction is made between the supplies needed for the equipment being used (fuel and ammo) and the supplies specifically required for the task (mines, explosives, wire). This list of supplies is limited to the supplies required for the task. Equipment will refuel and rearm automatically.

EN.TECH.SEGMENT (TEMPORARY ENTITY) - the smallest measurable part of a job using this technique. The use of this entity allows many variations in how a job is done and how equipment and supplies are used.

EN.SEGMENT.NUMBER (ATTRIBUTE) - the sequence number of this segment. The segments are numbered consecutively by this attribute to be used as a reference.

EN.END.SEGMENT.EFFECT (ATTRIBUTE) - the fraction of the total effect of this technique that is complete at the end of this segment. If the job is interrupted at this point, the effect of the job is adjusted according to this number.

EN.INTRA.SEGMENT.EFFECT (ATTRIBUTE) - a defined-to-mean value which indicates the interpolation factor for computing the effect of completion of the job to this point, given the end effects of the last segment completed and the segment in progress. The define-to-means are:

..NO.CHANGE	TO MEAN 1
..LINEAR	TO MEAN 2
..SKEW.LEFT	TO MEAN 3
..SKEW.RIGHT	TO MEAN 4

If this value is 1, the end effect of the last completed segment is used. For the other three values, the ratio of the time spent on the segment to the total time required for the segment is computed. If the intrasegment effect is 2, the interpolation factor is the computed ratio. If it is 3, the interpolation factor is the square root of the computed ratio. If it is 4, the interpolation factor is the square of the computed ratio.

EN.SEGMENT.DURATION (ATTRIBUTE) - the time in seconds for completion of this segment, given that all required equipment is present and conditions are ideal (good weather, daytime, no combat). The actual job duration will be computed by adjusting this duration when equipment levels fall, equipment is damaged, or any of the factors below change. The following tasks must have nonzero segment durations and may not have segment lengths determined by a rate:

..EMPLACE.OBSTACLE.COMPLEX
..BREACH.OBSTACLE.COMPLEX
..LINE.OBSTACLE.BREACH
..IMPROVE.LINE.BREACH
..REPAIR.ROAD.CRATER
..BRIDGE.DEMOLITION
..CRATER.ROAD
..PREPARE.POSITION
..BUILD.TRAIL

EN.SEGMENT.RATE (ATTRIBUTE) - the rate at which this segment is completed, given that the associated job has a nonzero size. If duration is zero, then the length of the segment will be computed by dividing the size of the job by this rate. The units of measure for the rate vary with the task type: mine/hour for minefield emplacement and clearing, kilometers of depth/hour for minefield breaching, kilometer/hour for linear obstacle emplacement and road maintenance.

EN.SEGMENT.VISIBILITY.FACTOR (ATTRIBUTE) - a multiplier which adjusts the segment duration to account for performance of the task at night.

EN.SEGMENT.WEATHER.FACTOR (ATTRIBUTE) - a multiplier which adjusts the segment duration to account for a drop of one level in weather (GOOD to FAIR, FAIR to POOR).

EN.SEGMENT.COMBAT.FACTOR (ATTRIBUTE) - a multiplier which adjusts the segment duration to account for working while under fire.

EN.END.SEGMENT.ACTION (ATTRIBUTE) - a defined-to-mean value which indicates one of three actions to take at the end of the segment. The defined-to means are:

..CONTINUE	TO MEAN 1
..BREAK.POINT	TO MEAN 2
..END	TO MEAN 3

EN.SET.OF.TECH.SEGMENTS (SET MEMBER) - the list of segments into which this technique is divided.

EN.EQUIPMENT.USE.SEGMENT (TEMPORARY ENTITY) - an entity created to allow the EN.REQ. EQUIPMENT to keep a list of segments in which it is used. Input data specifies in which segments (by number) each type of required equipment is used.

EN.EQ.USE.SEGMENT.NBR (ATTRIBUTE) - the segment number of one of the segments in which the equipment is used.

EN.SET.OF.EQUIPMENT.USE.SEGMENTS (SET MEMBER) - a list of the segments of a technique which use the required equipment type which owns this set.

EN.REQ.EQUIPMENT (TEMPORARY ENTITY) - an entity created to keep track of a type and amount of equipment required for the technique.

EN.EQ.PROTO (ATTRIBUTE) - the index for the asset prototype of this required equipment.

EN.EQ.AMOUNT (ATTRIBUTE) - the preferred number of pieces of equipment of this type that would be used for this technique. The amount of time it takes to do a job with this technique is based on this number.

EN.EQ.MINIMUM (ATTRIBUTE) - the minimum number of pieces of equipment of this type which must be present for a job using this technique to start or to continue. As the number of available pieces drops from the EN.EQ.AMOUNT to the EN.EQ.MINIMUM, the durations of the segments in which this equipment is used increase. If the level of equipment falls below this minimum, the job is discontinued.

EN.EQ.USE (ATTRIBUTE) - a defined-to-mean which indicates whether the equipment is ..PERMANENT or ..RETRIEVABLE. A retrievable asset does not return from the job site and is essentially consumed unless a later decision is made to retrieve it. In the case of bridging equipment, assets may be retrieved by bridge demolition (an external event) or by improving the bridge (an internally generated task).

EN.EQ.PREP.TIME (ATTRIBUTE) - the amount of time in seconds required to prepare this piece of equipment at the engineer HQ before it can leave for the work site.

EN.SET.OF.EQUIPMENT.USE.SEGMENTS (SET OWNER) - the list of segments in which this equipment is used.

EN.SET.OF.REQ.EQUIPMENT (SET MEMBER) - the list of all of the required equipment for the given technique.

EN.REQ.SUPPLY (TEMPORARY ENTITY) - an entity created to keep track of the supplies required for a job using this technique.

EN.SUPPLY.PROTO (ATTRIBUTE) - the supply prototype from the LO module.

EN.SUPPLY.AMOUNT (ATTRIBUTE) - the amount of the supply required.

EN.SUPPLY.UNIT.OF.MEASURE (ATTRIBUTE) - a supply consumption rate unit of measure for use with jobs that have an associated size. This attribute is the unit of measure associated with the required amount. The following formula is used to compute the supply requirement:

$$\text{REQUIRED SUPPLIES} = \frac{\text{JOB SIZE X SUPPLY AMOUNT}}{\text{SUPPLY UNIT OF MEASURE}}$$

EXAMPLES: For emplacing minefields, supply amount and unit of measure for mines are both 1, so that the required supplies equals the job size (the number of mines). For breaching minefields (job size is depth in kilometers) with line charges, if each line charge clears 150 meters, then supply amount is 1 and unit of measure is 0.15.

EN.SUPPLY.SEGMENT.USED (ATTRIBUTE) - the segment in which the supplies are effectively consumed. Interruption of a job before this point will mean that the supplies will return to the HQ unit.

EN.SET.OF.REQ.SUPPLIES (SET MEMBER) - the list of all of the required supplies for the given technique.

Engineer Jobs

EN.JOB (TEMPORARY ENTITY) - the master record keeper for specific engineer activity which has been assigned to an engineer HQ unit.

EN.JOB.TASK (ATTRIBUTE) - the defined-to-mean task type associated with the job.

EN.JOB.FEATURE (ATTRIBUTE) - the feature prototype associated with the job.

EN.JOB.FEATURE.PTR (ATTRIBUTE) - the pointer to the actual feature involved.

EN.JOB.TECHNIQUE (ATTRIBUTE) - the index of the technique by which the work is to be done. This may change until the job has been started, at which time the technique must remain constant.

EN.JOB.METHOD (ATTRIBUTE) - a defined-to-mean value which indicates how a technique is to be chosen for doing the job. ..PREFERRED.METHOD means that the first technique listed in the input data that is suitable for the job will be used. ..SHORTEST.TIME means that all usable techniques will be considered and the technique with the shortest completion time will be chosen. ..BEST.EFFECT means that all usable techniques will be considered and the technique with the most effective end result will be chosen. The default is ..PREFERRED.METHOD, but in dynamically generated jobs, the combat status of the requesting unit may prompt a switch to one of the other two methods.

EN.JOB.SIZE (ATTRIBUTE) - the size of the job if its task type is one that allows for rates in determining duration: number of mines for emplacing and clearing of minefields, length in kilometers for emplacing linear obstacles and maintaining roads, and depth in kilometers for breaching minefields.

EN.JOB.PRIORITY (ATTRIBUTE) - a numerical value which reflects the priority of this job, to be compared to other jobs in determining the order in which jobs should be done.

EN.JOB.MISSION.CREATION.TIME (ATTRIBUTE) - the clock time at which the original mission was created.

EN.JOB.ACTIVATION.TIME (ATTRIBUTE) - the clock time at which the first segment of the job was activated, i.e., when the first work team was formed to work on the first phase.

EN.JOB.EARLIEST.START.TIME (ATTRIBUTE) - the earliest clock time at which engineers may begin to work on this job.

EN.JOB.LATEST.COMPLETION.TIME (ATTRIBUTE) - the latest clock time for completion of the job. If the job cannot be completed by this time, it will be canceled as soon as that fact is recognized.

EN.JOB.CURRENT.SEGMENT.START.TIME (ATTRIBUTE) - the clock time at which the active segment started. The segments are sequential, so only one segment can be active at any given time.

EN.JOB.EST.DURATION (ATTRIBUTE) - an estimate in seconds of how much longer it should take to complete the job. This attribute is updated at each stage of job processing. It is used as an estimate in questions of job feasibility and in scheduling other events dependent on job completion (example: unit crossing of an enemy minefield).

EN.JOB.LAST.SEGMENT.COMPLETED (ATTRIBUTE) - the number of the last completed segment of the job.

EN.JOB.LAST.SEGMENT.COMPLETION.TIME (ATTRIBUTE) - the clock time at which the last completed segment of the job was done.

EN.JOB.LAST.SEGMENT.ASSIGNED (ATTRIBUTE) - the number of the last segment of the technique which has already been assigned to a work team. This attribute keeps track of where the HQ unit is in assigning portions of the job to work teams.

EN.JOB.X.LOCATION (ATTRIBUTE) - the x-coordinate of the work site.

EN.JOB.Y.LOCATION (ATTRIBUTE) - the y-coordinate of the work site.

EN.JOB.REQUESTING.UNIT (ATTRIBUTE) - the GG.UNIT for whom the work is being done, if one exists. Preplanned obstacles will have requesting units if the optional data item was supplied in input; defensive preparations and dynamically generated jobs will always have a requesting unit. This status of this unit is a factor in determining the job priority.

EN.JOB.HQ.UNIT (ATTRIBUTE) - the HQ unit which is responsible for this job and which has this job on its pending or active job list.

EN.JOB.MISSION (ATTRIBUTE) - the model-assigned mission number for this job. As jobs are created and destroyed, the entity address for the job may be repeated for several different jobs, so the entity address is not sufficient for distinguishing one job from another. Assigning a sequential mission number to each job allows output to identify jobs using the same entity address but different mission numbers as different jobs. Also, input data allows an "original mission number" for specifying when several jobs of one type will be done by one work team, but no requirement is made to use distinct engineer mission numbers in the different modules. For instance, both linear obstacle emplacement and minefield emplacement may have an engineer mission number 1. Because the EN module mixes all types of jobs on the same list, the original mission number cannot be used for this attribute and has no relationship to it.

EN.JOB.ORIGINAL.MISSION (ATTRIBUTE) - the engineer mission number assigned to this task in the input data for jobs that are a part of the countermobility plan. Otherwise, this attribute is zero.

EN.JOB.OC.PTR (ATTRIBUTE) - a pointer to the obstacle complex that this job involves if the job involves working on only one part of the obstacle complex. Otherwise, this attribute is zero.

EN.SET.OF.ACTIVE.PHASES (SET OWNER) - a set to keep track of which phases of a job have been assigned but not completed. Each phase of the job will be done by a different set of work teams. Because of grouping jobs by mission. several phases of a job may be active at once even though the progress on the job proceeds from one segment to the next in proper sequence.

EN.SET.OF.HQ.PENDING.JOBS (SET MEMBER)

EN.SET.OF.HQ.ACTIVE.JOBS (SET MEMBER) - the sets that keep track of the jobs assigned to a particular HQ unit. A job is either pending or active and is therefore on one of the above lists for the HQ unit which is responsible for it.

EN.ACTIVE.PHASE (TEMPORARY ENTITY) - an entity created to keep track of the parts of a job that have work teams assigned to them. The technique for the job divides the job into segments which are grouped together in phases according to the EN.END.SEGMENT.ACTION. Each phase is performed by a different set of work teams.

EN.PHASE.FIRST.SEGMENT (ATTRIBUTE) - the number of the segment with which this phase begins. This attribute is updated as the work proceeds and each first segment is completed.

EN.PHASE.LAST.SEGMENT (ATTRIBUTE) - the number of the segment with which this phase ends.

EN.PHASE.ASSET.PTR (ATTRIBUTE) - a pointer to a 1 by N.EN.ASSET.PROTO array whose nth element is the number of pieces of equipment of type n required during this phase of the job. This is the EN.EQ.AMOUNT for this equipment prototype. It is used as the comparison figure to determine the segment duration. If all equipment is present in the quantities indicated by this array, then the segment duration is taken from the input value. Otherwise, a new duration is computed.

EN.SET.OF.PHASE.WT.UNITS (SET OWNER) - a set to keep track of the WT units created to work on this phase of the job.

EN.SET.OF.ACTIVE.PHASES (SET MEMBER) - a list of active phases of the job.

EN.PHASE.WT.UNIT (TEMPORARY ENTITY) - an entity created to keep track of the work teams assigned to this phase of the job. The work teams themselves are filed in the EN.SET.OF.PERFORMING.UNITS of the HQ unit which owns the job.

EN.PHASE.WT.PTR (ATTRIBUTE) - the pointer to the EN.WT.UNIT which is assigned to this phase.

EN.SET.OF.PHASE.WT.UNITS (SET MEMBER) - a set to keep track of the work teams assigned to a particular phase of a job. The set is owned by an EN.ACTIVE.PHASE.

EN.WT.JOB (TEMPORARY ENTITY) - an entity created to allow each work team for a job to keep track of the job.

EN.WT.JOB.PTR (ATTRIBUTE) - a pointer to the corresponding EN.JOB.

EN.WT.JOB.PHASE (ATTRIBUTE) - a pointer to the EN.ACTIVE.PHASE to which this work team is assigned.

EN.SET.OF.WT.JOBS (SET MEMBER) - the list of jobs assigned to the work team.

EN.STAFF.JOB (TEMPORARY ENTITY) - an entity created to allow each staff unit to keep track of a job assigned to one of the HQ units in the staff's command chain. The staff unit is responsible for prioritizing the jobs in its entire command chain. It uses this entity to build a list of those jobs to use as a working set when determining priorities. An EN.STAFF.JOB is created for each job in the command chain.

EN.SJ.RESPONSIBLE.UNIT (ATTRIBUTE) - the HQ unit responsible for the corresponding EN.JOB.

EN.SJ.PTR (ATTRIBUTE) - a pointer to the corresponding EN.JOB.

EN.SJ.ACTIVE.INDICATOR (ATTRIBUTE) - a flag indicating whether the corresponding job is active or not. This attribute is 0 if the job is pending and 1 if it is active.

EN.SJ.PRIORITY.LEVEL (ATTRIBUTE) - the priority level of the job which is an assigned real number and which determines the order in which pending jobs are processed.

EN.SJ.READY.FLAG (ATTRIBUTE) - a flag used in the prioritization process to force jobs overtaken by events to be pushed to the bottom of the job processing list.

EN.SET.OF.STAFF.JOBS (SET MEMBER) - the list of all of the jobs, both pending and active, for which units in this staff's command chain are responsible.

Combat Trails

EN.COMBAT.TRAIL (TEMPORARY ENTITY) - an entity created to keep track of an engineer terrain feature which alters the trafficability of maneuver units travelling through its grid cell.

EN.CT.SIDE (ATTRIBUTE) - the side constructing the combat trail.

EN.CT.X.GRID (ATTRIBUTE) - the x-grid number of the grid cell containing the combat trail.

EN.CT.Y.GRID (ATTRIBUTE) - the y-grid number of the grid cell containing the combat trail.

EN.CT.EFFECT (ATTRIBUTE) - the number of levels of improvement in trafficability afforded by the combat trail. This number is derived from the relative effect of the technique used to construct the trail and can be no larger than one less than the original trafficability level of the grid.

Define-to-Means for Engineers

The two extreme levels of position preparation:

```
DEFINE ..UNPREPARED              TO MEAN  0.0
DEFINE ..FULLY.PREPARED          TO MEAN  1.0
```

The set of values used in producing a defensive position report for the engineer history file:

```
DEFINE ..CREATE.DEF.POS                TO MEAN  1
DEFINE ..UPDATE.DEF.POS                TO MEAN  2
DEFINE ..OCCUPY.DEF.POS                TO MEAN  3
DEFINE ..LEAVE.DEF.POS                 TO MEAN  4
DEFINE ..DESTROY.ENEMY.DEF.POS         TO MEAN  5
DEFINE ..SAVED.BY.DEF.POS              TO MEAN  6
```

The types of tasks engineers are capable of performing:

```
DEFINE ..MINEFIELD.BREACH              TO MEAN  1
DEFINE ..MINEFIELD.CLEAR               TO MEAN  2
DEFINE ..LINE.OBSTACLE.BREACH          TO MEAN  3
DEFINE ..BREACH.OBSTACLE.COMPLEX       TO MEAN  4
DEFINE ..IMPROVE.LINE.BREACH           TO MEAN  5
DEFINE ..REPAIR.ROAD.CRATER            TO MEAN  6
DEFINE ..BUILD.TRAIL                   TO MEAN  7
DEFINE ..EMPLACE.MINEFIELD             TO MEAN  8
DEFINE ..EMPLACE.LINE.OBSTACLE         TO MEAN  9
DEFINE ..EMPLACE.OBSTACLE.COMPLEX      TO MEAN 10
DEFINE ..BRIDGE.DEMOLITION             TO MEAN 11
DEFINE ..PREPARE.POSITION              TO MEAN 12
DEFINE ..CRATER.ROAD                   TO MEAN 13
DEFINE ..MAINTAIN.ROAD                 TO MEAN 14
```

The number of tasks engineers are capable of performing:

```
DEFINE ..NUMBER.TASKS                  TO MEAN 14
```

The set of values for EN.ASSET.STATUS:

```
DEFINE ..AVAILABLE                     TO MEAN  1
DEFINE ..PRE.TASK                      TO MEAN  2
DEFINE ..IN.TRANSIT                    TO MEAN  3
DEFINE ..WAITING                       TO MEAN  4
DEFINE ..WORKING                       TO MEAN  5
DEFINE ..IN.REST.PERIOD                TO MEAN  6
DEFINE ..DAMAGED                       TO MEAN  7
DEFINE ..LEFT.AT.SITE                  TO MEAN  8
```

The set of values for EN.TECH.ORGANIZATION.PT:

```
DEFINE ..AT.HQ                         TO MEAN  0
DEFINE ..AT.SITE                       TO MEAN  1
```

The set of values for EN.INTRA.SEGMENT.EFFECT:

```
DEFINE ..NO.CHANGE                   TO MEAN  1
DEFINE ..LINEAR                      TO MEAN  2
DEFINE ..SKEW.LEFT                   TO MEAN  3
DEFINE ..SKEW.RIGHT                  TO MEAN  4
```

The set of values for EN.END.OF.SEGMENT.ACTION:

```
DEFINE ..CONTINUE                    TO MEAN  1
DEFINE ..BREAK.POINT                 TO MEAN  2
DEFINE ..END                         TO MEAN  3
```

The set of values for EN.EQ.USE:

```
DEFINE ..PERMANENT                   TO MEAN  1
DEFINE ..RETRIEVABLE                 TO MEAN  2
```

The set of values used in classifying engineer task types into mission areas:

```
DEFINE ..MOBILITY                    TO MEAN  1
DEFINE ..COUNTERMOBILITY             TO MEAN  2
DEFINE ..SURVIVABILITY               TO MEAN  3
DEFINE ..GENERAL.ENG                 TO MEAN  4
```

The set of values used in producing a job report for the engineer history file:

```
DEFINE ..CREATE.JOB                  TO MEAN  1
DEFINE ..ACTIVATE.PHASE              TO MEAN  2
DEFINE ..BEGIN.SEGMENT               TO MEAN  3
DEFINE ..ADJUST.SEGMENT.DURATION     TO MEAN  4
DEFINE ..END.SEGMENT                 TO MEAN  5
DEFINE ..COMPLETE.PHASE              TO MEAN  6
DEFINE ..COMPLETE.JOB                TO MEAN  7
DEFINE ..COMPLETE.MISSION            TO MEAN  8
DEFINE ..PREEMPT.EQUIPMENT           TO MEAN  9
DEFINE ..DISCONTINUE.ATTRITION       TO MEAN 10
DEFINE ..DISCONTINUE.LATE.FINISH     TO MEAN 11
DEFINE ..DISCONTINUE.RELATED.JOBS    TO MEAN 12
```

The set of values for EN.MISSION.METHOD and EN.JOB.METHOD:

```
DEFINE ..PREFERRED.METHOD            TO MEAN  0
DEFINE ..SHORTEST.TIME               TO MEAN  1
DEFINE ..BEST.EFFECT                 TO MEAN  2
```

6 SUMMARY OF ROUTINES

Overview

The routines and events of the EN module are listed below in alphabetical order with given and yielded arguments and with brief descriptions. Each routine/event name follows the VIC convention of beginning with the two letter module abbreviation (EN); each name describes the routine/event function as closely as possible. Argument names follow the VIC convention for naming local variables, beginning with a "." and using a "." to separate words within the name.

Description

ROUTINE EN.ADD.AVAILABLE.ASSETS
GIVEN .UNIT, .ASSET.ARRAY, .PRIORITY

This recursive routine cycles through a staff unit and all of its HQ subordinates and their HQ subordinates down the command chain, adding up all of the engineer equipment available at the HQ and at its work teams working on lower priority jobs so that .ASSET.ARRAY indicates all of the assets available within that command chain to work on a job of the given .PRIORITY.

ROUTINE EN.ADD.JOBS.TO.LIST
GIVEN .STAFF, .UNIT

This recursive routine cycles through all of the .UNITs in the given .STAFF's command chain adding a job to the staff unit's set of jobs for each active and pending job assigned to .UNIT. All units subordinate to a staff unit share equipment, so prioritizing and processing of jobs is controlled at the staff level. The list that is constructed here is the working list for that process. Individual units have lists of pending and active jobs which track job history; these lists are used here to construct the staff's working list.

ROUTINE EN.ARE.WTS.COLLOCATED
GIVEN .WT1, .WT2
YIELDING .ANSWER

This routine determines whether two work teams, .WT1 and .WT2, are at the same location, returning an .ANSWER of ..YES if the center locations of the two units are closer together than the maximum of their deployment radii. Otherwise .ANSWER is ..NO.

ROUTINE EN.ASSESS.DIRECT.ENGINEER.SUPPORT
GIVEN .UNIT, .MISSION, .FEATURE, .X.LOC, .Y.LOC, .SIZE

This routine is the key to the construction of the set of engineer units capable of performing the given mission. It searches the given nonengineer unit's subordinate list for staff engineers capable

of performing the given mission sometime during the course of the battle. Capability is determined by resources available to the staff and tactical areas assigned to the staff. This routine adds each capable unit to the set of possible units for the mission.

ROUTINE EN.ASSESS.ENGINEER.CAPABILITY
GIVEN .UNIT, .CURRENT.DELAY, .TASK, .OC.PTR, .FEATURE,
.METHOD, .NR.OBS
YIELDING .TOTAL.DURATION, .EN.OBS

This routine determines whether engineers currently have the capability to breach the given obstacle in less time than .CURRENT.DELAY, which is the encountering .UNIT's computed delay if the breach is done without engineer help. If engineer work is already in progress, the job duration is returned as the difference of the expected completion time and the current time. If no work is already in progress, both explicit and implicit capabilities are checked. If either work effort will produce a faster breach, the fastest method is chosen. In this case, .UNIT is assessed the shorter delay, and this routine generates the actual engineer work. This routine handles both minefields and obstacle complexes. With obstacle complexes, the feature may represent a number of minefields or line obstacles. The given unit's actual encounter with the complex determines the number of obstacles seen (.NR.OBS), and this routine must determine the number of such obstacles engineers have the resources to breach concurrently (.EN.OBS).

ROUTINE EN.ASSIGN.ENGINEER.JOB
GIVEN .SIDE, .MISSION, .FEATURE, .X.LOC, .Y.LOC, .SIZE
YIELDING .HQ, .TECHNIQUE

This routine handles the search for an engineer HQ unit on the given .SIDE to be responsible for the given .MISSION. If the mission has a requesting unit, this routine first determines if the requesting unit is capable of doing the job itself. Then that unit's direct engineer support is checked. Then the routine recursively checks the direct engineer support in the unit's chain of command. At each level, all capable staff units are added to the mission's possible units set. If that set remains empty or if the mission had no requestor, the search expands to consider all engineer staff units. Having established the set of possible engineer staff units, the process looks through the set for a staff unit whose current tactical area contains the mission, choosing the staff with the smallest job load if more than one qualifies. After determining the staff unit, it searches the staff's command chain for the HQ at the command level specified by the chosen technique and closest to the job site. The given .METHOD indicates the type of technique being sought, with the usual case of taking the first technique found in the input data, but allowing for taking the technique with the shortest duration or the best effect.

EVENT EN.BRIDGE.DEMOLITION

This event processes bridge preparation and demolition requested in an external file in the following format:

EN.BRIDGE.DEMOLITION DD HH MM X Y RANGE FLAG

where the time for the event is given in DHM format; X, Y, and range specify the location of the center and radius of a circle within which all bridges are to be either prepared for demolition or demolished. FLAG may be:

XB, XR - remove all bridges in the given area without engineer assistance and for the given side now

EB, ER - have engineers on the appropriate side prepare all bridges in the specified area for demolition; bridges remain when work is complete

DB, DR - remove each bridge in the specified area that is already prepared for demolition by this side; have engineers on this side prepare and demolish as soon as possible any unprepared bridges in the specified area

U followed by a GG unit name - with this option, the named unit becomes the destroyer of each prepared bridge in the specified area when the unit crosses the bridge.

This event may be used to retrieve assault bridging. If a bridge containing engineer assets is destroyed by the side owning the assets, then the assets are returned to the owning engineer HQ unit.

ROUTINE EN.BEGIN.CURRENT.JOB.SEGMENT
GIVEN .WT

This routine is called when the given engineer work team arrives at the work site of the first job on its job list. The routine checks to see if the next segment to be done on the job has been assigned to this unit and, if so, computes how long the segment will take given current equipment levels and conditions. If everything is in order, the end of the next segment is scheduled here. Work teams can arrive at the job site before earlier phases have been completed, and equipment may have to come from several locations to work on one segment. If the work team is assigned to a phase of the job that is not ready to begin or if all of the equipment for this segment has not arrived, this routine does nothing; the completion of the preceding phase or the arrival of the needed equipment will trigger the work team to begin work.

ROUTINE EN.BUILD.STAFF.ARRAYS
GIVEN .STAFF

This routine cycles through the techniques for each task and feature combination, comparing the total undamaged asset holdings of .STAFF and all of its engineer HQ subordinates with the required equipment for the technique to determine whether or not this command chain can use the technique. This includes all techniques possible for this unit and its subordinates using any combination of equipment available to the group as a whole. The resulting technique array owned by the staff unit indicates the first technique for that task and feature combination that the staff can use. The array element is zero if the staff has no technique for that task and feature combination.

ROUTINE EN.BUILD.WORK.UNIT.AT.HQ
GIVEN .JOB, .PHASE, .CONTRIBUTING.HQ, .NEEDED.ASSET, .IN.FLAG
YIELDING .FLAG

This recursive routine moves through .CONTRIBUTING.HQ and all of the HQ units subordinate to it searching for the engineer assets required for the given .PHASE of the given .JOB. The array .NEEDED.ASSET indicates what is still needed. As long as .FLAG (.IN.FLAG) is ..YES the search continues. .FLAG is ..NO when the array is 0. At each .CONTRIBUTING.HQ, if needed equipment is found, it is transferred to a newly created work team, which will move it to the job. If LO is being played, each .CONTRIBUTING.HQ gives its equipment the fuel and ammunition required for the equipment. The first work team created during this process receives the required job supplies from the responsible HQ and carries them to the work site.

ROUTINE EN.BUILD.WORK.UNIT.AT.WT
GIVEN .JOB, .PHASE, .CONTRIBUTING.HQ, .NEEDED.ASSET, .IN.FLAG
YIELDING .FLAG

This recursive routine moves down the command chain from .CONTRIBUTING.HQ searching for equipment needed for the given .PHASE of the given .JOB held by work teams working on jobs of lower priority. If needed equipment is found, a new work team is created and the equipment is transferred to it to move to the new job site. If the preempted work team is currently using the equipment, the equipment leaves at the end of the current segment. The array .NEEDED.ASSET keeps track of what equipment is still being sought. The search continues until .FLAG (.IN.FLAG) is ..NO or the end of the chain is reached. Preprocessing guarantees that the minimum amount of required equipment will be found, and the search method guarantees that as much of the preferred amount that is available will be sent.

ROUTINE EN.CALCULATE.SEGMENT.DURATION
GIVEN .TECHNIQUE, .SEGMENT, .OWNED.EQ.ARRAY, .SIZE
YIELDING .DURATION, .PREP.TIME

The purpose of this routine is to calculate the .SEGMENT duration for a job of the given .SIZE using the given .TECHNIQUE if the performing unit can use the amount of equipment represented in the .OWNED.EQ.ARRAY. The timing factors given in the engineer input data are for segments using the preferred amount of required equipment. The duration must be adjusted upward as the amount of available equipment drops, with the required equipment minimum specifying the threshold at which the work cannot be done.

ROUTINE EN.CALCULATE.NEW.FINISH.TIME
GIVEN .WT, .TECHNIQUE, .SEGMENT
YIELDING .DURATION

This routine computes the .DURATION of a .SEGMENT of a job by comparing the assets of .WT unit doing the work with the equipment required for this segment of the given .TECHNIQUE. This routine is only used when the work team has less equipment than the preferred amount for the technique. If the level of any of the required equipment is below the minimum specified by input

data, then the duration is set high enough to cause the job to be discontinued. The asset array expresses in integer mode the number of pieces of equipment owned, while the EN.ASSET.UNDAMAGED.PORTION or the EN.WT.ASSET.PORTION of each asset expresses the usable portion of that asset. Work time is degraded if the equipment is not at 100 percent.

ROUTINE EN.CHECK.ON.WT.PROGRESS
GIVEN .SIDE

This routine periodically checks every work team to see if it can no longer get to its job in time to complete work. If such a work team is found and its equipment is not critical to the job, it is sent home and the job continues. If the job cannot be completed without that work team's equipment, the job is discontinued.

ROUTINE EN.CHECK.FOR.SUPPLY.TYPES.IN.INVENTORY
GIVEN .HQ.GG, .TECHNIQUE
YIELDING .ANSWER

This routine compares the supply list for a .TECHNIQUE with the supply inventory for an HQ unit and returns ..YES in .ANSWER if the unit receives each type of required supply. This routine is used as a preliminary check when assigning a mission. The given unit must actually receive the required supply types if it is to use the given technique. The required supplies do not have to be on hand at the unit to make the assignment of a mission using this technique.

ROUTINE EN.CHECK.FOR.AVAILABILITY.OF.JOB.SUPPLIES
GIVEN .UNIT, .TECHNIQUE, .START, .SIZE
YIELDING .ANSWER

This routine compares the supply list for a .TECHNIQUE with the supply inventory for a GG unit and returns ..YES in .ANSWER if the unit has the required amount of each type of supply on hand to be expended during the job segments from the .START segment to the end of the phase. .SIZE is the job size.

ROUTINE EN.CLOSE.OUT.JOB.PHASE
GIVEN .WT

This routine does the bookkeeping that is needed when a work team completes a phase of a job. The job attributes must be updated, and the work team sent to either its next job or home if it is finished.

ROUTINE EN.COMPARE.OWNED.TO.REQUIRED
GIVEN .UNIT.ASSET.ARRAY, .JB.ASSET.ARRAY
YIELDING .ANSWER

This routine compares owned assets with required assets and is used in two ways: to decide which techniques a staff's command chain can use by comparing the total of its assets with the minimum assets required by the technique and to decide whether the assets owned by a command chain and available or working on a lower priority job are sufficient to meet the minimum needs of the current phase of a mission. In both instances, .UNIT.ASSET.ARRAY indicates what is owned, and .JB.ASSET.ARRAY indicates the minimum needed. If any element of the .JB.ASSET.ARRAY is larger than the corresponding element of .UNIT.ASSET.ARRAY, then .ANSWER is ..NO; otherwise, ..YES.

ROUTINE EN.COMPUTE.DISTANCE.TO.JOB
GIVEN .UNIT, .JOB
YIELDING .DISTANCE

This utility routine computes the .DISTANCE between a given GG.UNIT called .UNIT and a given .JOB.

ROUTINE EN.COMPUTE.ENGINEERS.TRUE.STRENGTH
GIVEN .UNIT, .WG
YIELDING .STRENGTH

This routine determines the weapon group strength of an engineer unit in terms of what the unit owns versus what the unit presently has on hand, which is the value of FL.WG.STRENGTH. This routine is used for determining supply and maintenance needs.

ROUTINE EN.COMPUTE.TRAVEL.TIME.BETWEEN.TWO.POINTS
GIVEN .X.A, .Y.A, .X.B, .Y.B, .UNIT, .TIME.IN
YIELDING .X.PLAN, .Y.PLAN, .TRAVEL.TIME

This routine computes the travel time for travel from A to B if the time is within the planning time .TIME.IN; otherwise, it returns the location when the .TRAVEL.TIME equals the planning time .TIME.IN. This routine is used to determine the location of a planning point for an obstacle by backing away from the obstacle encounter point along the path to a point where the unit's travel time to the obstacle will be equal to the input value of the engineer planning time. If the unit's present location is encountered before the planning point can be placed, the planning point is filed at the unit's present location.

ROUTINE EN.COMPUTE.WORK.TEAM.SPEED
GIVEN .WT.GG, .BASE.SPEED
YIELDING .SPEED

This routine computes the speed at which a work team can travel, subject to the speeds of the assets it owns. GG prototype data specifies a maximum day and night speed for engineer work teams. That .BASE.SPEED is degraded still further if the given work team owns engineer equipment which moves at an even slower pace.

ROUTINE EN.CONSTRUCT.A.MINEFIELD.MISSION
GIVEN .UNIT

This routine constructs a minefield mission in support of .UNIT, which is considered the requesting unit for this mission. This is a holdover from VIC 2.0, which allowed minefield missions to be generated by decision tables.

ROUTINE EN.CONSTRUCT.AN.OBSTACLE.MISSION
GIVEN .UNIT

This routine is a holdover from VIC 2.0, where it was envisioned that a new obstacle mission would be generated by decision table action. As the current EN module was developed, this approach became less feasible and has not been implemented. This stub exists because of the possibility of a decision table call.

ROUTINE EN.CREATE.A.DEFENSIVE.POSITION
GIVEN .UNIT, .PATH.POINT
YIELDING .DP

This routine creates a defensive position for the given unit at the given path point location and files it in the set of defensive positions.

ROUTINE EN.CREATE.A.MISSION.REQUEST
GIVEN .TASK, .X.LOC, .Y.LOC, .UNIT, .TIME, .FEATURE, .METHOD

This routine creates an engineer mission for the given data. The assumption is that this is a dynamically generated new mission involving only one feature and that it should be processed immediately. Before the mission is created and filed, this routine checks the current mission and job lists for the side to be sure that this is not a duplicate request.

ROUTINE EN.CREATE.A.PLANNING.PATH.POINT
GIVEN .X.INTER, .Y.INTER, .FS, .X.LOC, .Y.LOC, .UNIT, .TASK, .TRAVEL.TIME
YIELDING .NEW.PT

This routine creates a planning path point with the given parameter values.

ROUTINE EN.CREATE.A.WORK.UNIT
GIVEN .HQ, .X.LOCATION, .Y.LOCATION, .PRIORITY
YIELDING .WT

This routine retrieves an engineer work team .WT from the free pool and sets its attributes. .HQ is .WT's superior, and .WT's location is .X.LOCATION, .Y.LOCATION. .PRIORITY is the priority of the job for which the work team is being created and is used in filing the work team in priority order in .HQ's set of performing units.

ROUTINE EN.CREATE.HQ.UNITS

This routine, used during the initialization process, builds a list of engineer HQ units for each side and establishes their equipment inventories and asset arrays from the FL weapon group data. It also builds the lists of engineer staff units, which are HQ units with nonengineer GM.UN.SUPERIORs.

ROUTINE EN.CREATE.MISSION.FOR.A.PLANNING.POINT
GIVEN .UNIT, .PLANNING.PT

This routine creates an engineer mission for the given unit at the given planning point.

ROUTINE EN.DELAY.ENGINEERS.IF.AT.TASK.SITE
GIVEN .UNIT, .DELAY.TIME

This routine resets the time an engineer work team spends at the task site given .DELAY.TIME imposed by current artillery or air activity.

ROUTINE EN.DETERMINE.JOB.LOAD
GIVEN .UNIT, .OLD.TOTAL
YIELDING .NEW.TOTAL

This recursive routine cycles through a staff unit and all of its HQ subordinates and their subordinates down the command chain, adding up all of the engineer jobs, both pending and active, to return the total number of jobs assigned to this command chain. It is used in the assignment process to help distribute the work load evenly among the qualifying staffs.

ROUTINE EN.DETERMINE.MINIMUM.EQUIPMENT.ARRAYS
GIVEN .TECHNIQUE, .FIRST, .LAST, .ASSET.ARRAY

This routine determines the minimum equipment required for .TECHNIQUE for the phase of work from segment .FIRST to segment .LAST and builds the minimum equipment array. This array determines the amount of equipment a unit must currently be able to give to a job using this technique in order to start the given phase.

ROUTINE EN.DETERMINE.REQUIRED.EQUIPMENT.ARRAYS
GIVEN .TECHNIQUE, .FIRST, .LAST, .ASSET.ARRAY

This routine retrieves the required equipment information for .TECHNIQUE for the phase of work from segment .FIRST to segment .LAST and builds the preferred equipment array. This array determines the maximum amount of equipment a unit will attempt to assign to the given phase of a job using this technique.

ROUTINE EN.DETERMINE.TRAIL.LOCATION
GIVEN .UNIT, .X.GRID, .Y.GRID
YIELDING .X2, .Y2

This routine finds the midpoint of the grid line closest to the unit responsible for building a trail through this grid. The job site for building a combat trail is external to the grid containing the trail to avoid having the work team suffer from poor trafficability within the grid itself and having too much delay assessed.

ROUTINE EN.DETERMINE.WT.UNIT.PATH
GIVEN .WT, .X.DESTINATION, .Y.DESTINATION, .DEST.TYPE,
.START.TIME

This routine builds the path of .WT from its present location to its .DESTINATION. This is a universal path builder, so code needs to be as generic as possible. .DEST.TYPE specifies the GG.PP.TYPE of the destination, which can be either a base point or a task point. .START.TIME specifies when the move is to begin. Engineer work teams actually generate their paths dynamically, so some care must be taken to route them around known obstacles and to bridges for river crossings. Work teams sent to breach an obstacle do not suffer delays from that obstacle.

ROUTINE EN.DISCONTINUE.ENGINEER.MISSION
GIVEN .WT.GG

This routine does the bookkeeping necessary to cancel the mission of the work team corresponding to the GG.UNIT .WT.GG and to send the work team back to its base. The work team is withdrawn from the phase of each job on its list, and an assessment is made as to whether the jobs can continue without the contribution of this work team.

ROUTINE EN.DISSOLVE.WT.UNIT
GIVEN .WT

When a work team has reached its final destination and has transferred all of its assets to another unit, this routine takes care of the bookkeeping involved in emptying all of the unit's sets and refiling it in the set of free units.

ROUTINE EN.DOES.THIS.COMMAND.HAVE.SUPPLIES
GIVEN .UNIT, .TECHNIQUE
YIELDING .ANSWER

This recursive routine cycles through a staff unit and all of its subordinates looking for a unit whose supply inventory contains all of the supply types required to use the given .TECHNIQUE.

EVENT EN.END.OF.JOB.SEGMENT
GIVEN .WT

This routine registers the completion of the current segment of the first job on .WT's job list and determines the work team's next course of action.

EVENT EN.END.OF.REST.PERIOD
GIVEN .ASSET

This routine resets an asset's attributes after its rest period so that the asset may be assigned to jobs again.

ROUTINE EN.FILE.A.PLANNING.PATH.POINT
GIVEN .UNIT, .OB.PP, .X.INTER, .Y.INTER, .TASK
YIELDING .NEW.PT

This routine searches back for an amount of time equal to the engineer planning time to find the location of the point at which to plan an engineer mission for the obstacle, creates a planning path point and files it in the .UNIT's path, and creates a mission for breaching the obstacle. The mission is held by the path point and not processed until the unit has its path point arrival at the planning point.

ROUTINE EN.FILE.FREE.UNIT.IN.POOL
GIVEN .WT

This routine removes an engineer work team from active status and files it in the free pool. .WT is the work team. Necessary GG.UNIT attributes are set so that the associated GG.UNIT is effectively removed from action.

ROUTINE EN.FILE.FREE.WEAPON.GROUP
GIVEN .WPN.GROUP

This routine clears the attributes of a weapon group and files it in the free set to be used again when a work team needs a weapon group to hold assets being transferred to it.

ROUTINE EN.FIND.BREACH.FOR.LINE.OBSTACLE
GIVEN .UNIT, .FS, .X.INTER, .Y.INTER
YIELDING .FLAG

This routine determines whether or not the given feature segment has a breach point within the unit's deployment or within 5 km of the given path intersection with the line feature and returns ..YES or ..NO in .FLAG.

ROUTINE EN.FIND.BYPASS.FOR.LINE.OBSTACLE
GIVEN .UNIT, .RADIUS, .FS, .X.INTER, .Y.INTER
YIELDING .X.C, .Y.C

This routine finds the bypass point C for the given line obstacle .FS.

ROUTINE EN.FIND.ENGINEER.HQ
GIVEN .UNIT
YIELDING .HQ

This routine links a GG.UNIT of type ..ENGINEER.UNIT with its associated .HQ unit, returning 0 if the unit is not in the set of engineer HQ units.

ROUTINE EN.FIND.OBSTACLE.BETWEEN.TWO.POINTS
GIVEN .UNIT, .UNIT.RADIUS, .X.A, .Y.A, .X.B, .Y.B, .FEATURE
YIELDING .X.C, .Y.C

This routine is a key part of the dynamic generation of an engineer work team path which avoids known obstacles. It finds the first obstacle which is either a line feature or a minefield located between A and B, and finds a point C by which the unit is able to bypass the obstacle. A is the starting point, and B is the destination point of the unit. If there is no obstacle between A and B, it returns B. .FEATURE is the feature that requires no bypass; it is the obstacle the work team is attempting to reach.

ROUTINE EN.FIND.SHORTEST.PATH.BETWEEN.TWO.POINTS
GIVEN .POINT.A, .POINT.B, .NEW.ROUTE, .FEATURE, .RADIUS

This routine recursively finds the shortest path from point A to point B.

ROUTINE EN.FIND.STAFF.ENGINEER
GIVEN .UNIT
YIELDING .STAFF

This routine links a GG.UNIT of type ..ENGINEER.UNIT to its associated .STAFF unit, returning 0 if the unit is not a staff engineer.

ROUTINE EN.FIND.STAFF.FOR.THIS.HQ
GIVEN .HQ
YIELDING .STAFF

This routine cycles up the command chain from .HQ until locating its .STAFF unit.

ROUTINE EN.FORM.INITIAL.WT.UNITS
GIVEN .THIS.JOB, .PHASE, .NEEDED.ASSET

This routine searches .THIS.JOB's responsible unit's command chain for the equipment needed for .PHASE. The .NEEDED.ASSET array indicates how much equipment is needed. The elements of this array approach zero as equipment is located. If all elements are zero, the desired equipment levels have been satisfied. Prior processing assures that the minimum amount of the required equipment will be found. The search method finds as much of the needed equipment as is available.

ROUTINE EN.GET.TASK.MAPPING
GIVEN .TASK
YIELDING .TASK.MAPPING

This utility routine associates an engineer mission category with each task type being modeled. .TASK is the define-to-mean number for the task, and .TASK.MAPPING is its define-to-mean category (MOBILITY, COUNTERMOBILITY, SURVIVABILITY, GENERAL ENGINEERING).

ROUTINE EN.GET.TASK.NAME
GIVEN .TASK
YIELDING .NAME

This utility routine associates a text name with each task type being modeled. .TASK is the define-to-mean number for the task, and .NAME is its text counterpart.

ROUTINE EN.GET.TASK.SPECIFICS.FOR.MISSION
GIVEN .MISSION
YIELDING .TASK.TYPE, .FEATURE.TYPE, .FEATURE, .X.LOC, .Y.LOC, .SIZE

This routine is the link between the very specific tasks requested of engineers and the processing of those tasks in a generic fashion. The key yielded arguments are derived for each specific task. Generic processing means that all jobs are handled by the same algorithms, using the above arguments to index into the technique array to determine how the job will be done and who will do it.

ROUTINE EN.HOW.MUCH.TIME.IS.NEEDED
GIVEN .TECHNIQUE, .JOB.SIZE, .SEGMENT
YIELDING .EST.DURATION

This routine totals the segment durations from .SEGMENT to the last segment for this .TECHNIQUE for a job of .JOB.SIZE and returns the estimated remaining duration. This is an estimate of the fastest completion time possible and is used as a cut-off time for canceling jobs that cannot possibly be done in time.

ROUTINE EN.HQ.DUMP
GIVEN .HQ

This routine produces a headquarters status report for one of the interactive menu options of EN.PRINT.CONTROL.

ROUTINE EN.INITIALIZE.TASK.TECH.ARRAY

This routine builds a ragged array dimensioned by .SIDE, .TASK, .FEATURE, and .TECH.NO so it is known immediately which techniques are available for each engineer effort as determined by the first three dimensions.

ROUTINE EN.IS.THIS.MISSION.INSIDE.TACTICAL.AREA
GIVEN .UNIT, .X.LOC, .Y.LOC
YIELDING .RESULT

This routine determines if .UNIT has the given site in its current or future tactical area. 1 is returned if the location is in the current tactical area; 2 if it is in a future tactical area; 0 if never.

ROUTINE EN.IS.THIS.NON.ENGINEER.CAPABLE
GIVEN .UNIT, .MISSION, .FEATURE, .SIZE
YIELDING .TECHNIQUE

This routine is called to determine whether or not a nonengineer .UNIT has the capability to perform the engineer .MISSION implicitly. It uses the technique data from the EN module, comparing the "engineer equipment" owned by the given unit with the equipment required for the mission. If the equipment is owned, this routine returns the best .TECHNIQUE, which is chosen based on the criteria determined by the mission's given method (most preferred, shortest time, most effective).

ROUTINE EN.IS.THIS.STAFF.UNIT.CAPABLE
GIVEN .STAFF, .TASK, .FEATURE, .X.LOC, .Y.LOC, .SIZE, .METHOD, .FLAG
YIELDING .TECHNIQUE

This routine is part of the process for determining the engineer unit to be responsible for assigning a work team to do a job of the given .TASK and .FEATURE at the given location .X.LOC,.Y.LOC. Each technique for the given task and feature is considered, checking whether the staff has the required equipment and receives the required supplies. If .FLAG is ..YES, then the minimum-distance-to-FEBA requirement of the technique is checked. Using the search method given by .METHOD, the best technique is determined and returned. If no technique is available that satisfies the conditions, .TECHNIQUE = 0.

ROUTINE EN.IS.WT.CRUCIAL.TO.PHASE.COMPLETION
GIVEN .WT, .WT.PHASE, .TECH
YIELDING .ANSWER

This routine determines whether or not the phase of a mission that the work team .WT is assigned can be completed without .WT. If .WT has equipment that will cause the equipment levels to fall below the minimum thresholds, then .ANSWER is ..YES.

ROUTINE EN.IS.WT.AT.BASE
GIVEN .WT
YIELDING .ANSWER

This utility routine determines whether the given .WT unit is at its base HQ location.

ROUTINE EN.IS.WT.AT.WORK.SITE
GIVEN .WT
YIELDING .ANSWER

This utility routine determines whether the given .WT unit is at the site of its first job.

ROUTINE EN.JOB.DUMP
GIVEN .JOB

This routine produces a job status report for one of the interactive menu options of EN.PRINT.CONTROL.

EVENT EN.JOB.UPDATE
GIVEN .SIDE

This event schedules itself after its initial scheduling by SS.SCHEDULE.INITIAL.EVENTS to periodically check the sets of engineer missions so that jobs can be created and processed. The mission processing interval of the engineer input data normally determines the amount of time to

elapse between successive occurrences of this event. Certain situations can cause this event to be rescheduled for a shorter interval.

ROUTINE EN.LIST.ASSET.DATA

This routine lists the engineer asset data during initialization.

ROUTINE EN.LIST.JOB.PRIORITY.DATA

This routine lists engineer job prioritization data during initialization.

ROUTINE EN.LIST.TECHNIQUE.DATA

This routine lists the engineer technique data during initialization.

ROUTINE EN.LOAD.AND.LIST.DEFENSIVE.POSITION.DATA

This routine loads and lists the defensive position data during initialization. It also establishes a defensive position prototype for each unit prototype and updates the position prototype of each defensive position created when the unit paths were read.

ROUTINE EN.LOAD.AND.LIST.ENGINEER.DATA

This routine sequences the loading of the engineer data during initialization.

ROUTINE EN.LOAD.ASSET.DATA

This routine loads the engineer asset data during initialization.

ROUTINE EN.LOAD.JOB.PRIORITY.DATA

This routine loads engineer job prioritization data during initialization. This data allows the scenario builder to influence how a staff engineer unit will prioritize the jobs within its command chain.

ROUTINE EN.LOAD.TECHNIQUE.DATA

This routine loads the engineer technique data, does the preprocessing of this data to allow quick searches, and builds the staff engineer asset and technique arrays.

ROUTINE EN.LOCATE.DEFENSIVE.POSITION.FOR.UNIT
GIVEN .UNIT, .CURRENT.POINT
YIELDING .DEF.POS

This routine finds a defensive position of the correct type for a unit at its given path point. If the path point was declared fully prepared by the input data or engineer preparation of the point was requested, then a defensive position already exists for this unit and is pointed to by the GG.PP.POINTER of the path point. Otherwise, a search is made for an unoccupied position of the right type for this unit.

ROUTINE EN.MISSION.DUMP
GIVEN .MISSION, .SIDE, .REQUESTED

This routine produces a mission status report for one of the interactive menu options of EN.PRINT.CONTROL if .REQUESTED is SS.THE.SCREEN and an output mission report if .REQUESTED is SS.ENGINEER.HISTORY.FILE.

ROUTINE EN.MOVE.DOWN.COMMAND.CHAIN
GIVEN .SUPERIOR, .TECH, .X.LOC, .Y.LOC
YIELDING .SUBORDINATE

This routine is used in the mission assignment process after the responsible staff unit has been determined. In successive loops, it moves down the command chain from .SUPERIOR looking for the engineer HQ subordinate closest to the job site with the site in its tactical area and, if LO is played, with the required supply types in its inventory. It returns the GG.UNIT number of such a subordinate if it exists; otherwise, 0.

ROUTINE EN.ORDER.ENGINEER.WORK
GIVEN .SIDE, .MISSION
YIELDING .OUTCOME, .HQ

This is the transition routine for moving an engineer mission to engineer jobs. If the work is impossible or unassignable, .OUTCOME indicates that and .HQ is 0. If the work involved in the mission is assignable, then this routine creates a job for each mission feature of the mission, sets the job attributes, and assigns the jobs to an engineer HQ unit, .HQ.

ROUTINE EN.PERFORM.IMPLICIT.TASKS
GIVEN .UNIT

This routine allows the given .UNIT to perform missions implicitly on its side's mission list. Such missions must have the given unit as their requestor, and the unit must own equipment required in the engineer technique data for doing the type of work indicated by the mission's task and feature type. Prior processing has determined the unit's capability for work and placed it in the set of possible units for the mission. This routine takes care of finding those missions which are currently

within the unit's radius of control, determining whether the required equipment is available, and scheduling work completion if it is.

ROUTINE EN.PLAN.FOR.OBSTACLE.INTERACTIONS
GIVEN .OB, .SIDE

This routine searches the paths of maneuver units on the given .SIDE for an encounter with the given obstacle, and, if found, files a planning path point for the unit. The routine is called when a new obstacle is introduced either interactively or through decision tables.

ROUTINE EN.PLAN.FOR.OBSTACLES.IN.UNIT.PATH
GIVEN .UNIT, .NULL.PP

This routine finds all the obstacles located on the .UNIT's path beginning at the .NULL.PP, which was created by the calling routine to mark the point in the path at which the search is to start and at which placement of the planning point must end if travel time is less than planning time. For each obstacle, it calls EN.PLAN.FOR.THIS.OBSTACLE to determine if a planning point is needed.

ROUTINE EN.PLAN.FOR.THIS.OBSTACLE
GIVEN .UNIT, .OB.PP
YIELDING .PLAN.PT

This routine determines if .UNIT needs to plan for the obstacle to be encountered at path point .OB.PP. If so, it calls EN.FILE.A.PLANNING.PATH.POINT to determine the planning point.

ROUTINE EN.PRINT.CONTROL

This routine provides the interactive gamer control mechanisms for engineer-related data.

ROUTINE EN.PRINT.FINAL.REPORT

This routine takes care of printing all necessary output to the engineer history file and the engineer setup file at the end of the simulation.

ROUTINE EN.PRINT.TASK.MENU

This routine displays the list of task names and the numbers corresponding to them.

ROUTINE EN.PRIORITIZE.INDIVIDUAL.JOB
GIVEN .SJ.JOB

This routine assigns a priority level to the given .JOB. Several factors are included in determining the priority, which is a real number: type of task, combat status of the requesting unit or supported unit, margin of time between how long the job will take and when it must be finished, the level of the unit requesting the mission compared to the level of the HQ unit responsible for the work, and how far away the job is along the path of the requesting unit.

ROUTINE EN.PRIORITIZE.JOB.LIST
GIVEN .STAFF

This routine prioritizes the entire list of engineer jobs for .STAFF engineer and its command chain and then processes the pending jobs by this prioritization. Each job is assigned a real number priority level which is used to order the staff job list. The priority level is updated frequently and the possibility exists for active jobs to be ranked lower than new pending jobs, in which case needed equipment could be preempted from a present assignment.

ROUTINE EN.PROCESS.JOB
GIVEN .THIS.JOB

This routine takes a job from the pending list, attempts to find the assets and supplies needed for its accomplishment, and, if found, creates the work teams needed to bring the equipment together and move it to the work site.

EVENT EN.PROCESS.JOB.LIST
GIVEN .STAFF, .PRIORITY.FLAG

This event is scheduled whenever a staff needs to evaluate and process the jobs assigned to its command chain. The priority flag indicates that the staff needs to rebuild and prioritize its job list. Each pending job is then processed in priority order to see if equipment is available. This event occurs when equipment becomes available, supplies have been delivered, new jobs have been assigned, or the supported units have changed path direction or combat role.

ROUTINE EN.PROCESS.MISSION.LISTS
GIVEN .SIDE

This routine periodically examines the mission list for each side, removing those missions which are assignable and converting them to jobs and discarding missions that are impossible because no technique exists for the mission's task and feature, no unit exists which can do the work, or the mission's required completion time has passed.

ROUTINE EN.PROCESS.THIS.MISSION.NOW
GIVEN .SIDE, .MISSION

This routine takes care of dynamically-generated missions which must be processed immediately. An attempt is made to assign the given mission to an HQ unit and to process the resulting job immediately if the assignment is made.

ROUTINE EN.READ.FEATURE.NAME
GIVEN .TASK.TYPE
YIELDING .FEATURE.TYPE

When technique data is input, this routine reads the feature name and associates it with the corresponding feature prototype number. This will vary by the task type of the technique, so several cases must be handled.

ROUTINE EN.RECONCILE.ASSET.COUNTS
GIVEN .UNIT, .REASON

This routine balances an engineer unit's weapons set with its engineer inventory when changes occur because of attrition or return-to-duty repairs. Engineer assets are also weapons, and the FL module sees them only as weapons when it computes attrition. In order to bring the engineer bookkeeping in line with the weapon counts, this routine must compare totals and adjust accordingly. .UNIT is the GG.UNIT number of the engineer unit. This routine is called whenever the FL.WG.STRENGTH of an engineer's weapon group changes.

ROUTINE EN.REFILE.ASSET.IN.INVENTORY
GIVEN .ASSET, .HQ

This routine takes care of removing an asset from its inventory list and refiling it at the bottom of the section of assets of its type. This process helps to spread work effort over all assets of a given type, leaving the most rested assets at the top of the list for pending assignment.

ROUTINE EN.REGISTER.EFFECT.OF.JOB
GIVEN .JOB, .COMPLETED.EFFECT

When a job being performed by an engineer unit has reached and completed its final segment or has been interrupted at an intermediate segment with the potential that no more work will be done on it, the effect of the work to date is registered by calling this routine, which in turn calls the generic EN.REGISTER.EFFECT used for implicit as well as explicit engineer activity. .COMPLETED.EFFECT is the portion completed and must be between 0 and 1.

EVENT EN.REGISTER.EFFECT.OF.NON.ENGINEER.JOB
GIVEN .TASK, .FEATURE, .TECH, .X, .Y, .UNIT, .SIZE, .PTR, .MISSION, .START.TIME, .END.TIME

A job being done implicitly by a nonengineer uses this event to schedule the completion of the work, with the possibility that the timing of the event can be changed if the unit is interrupted. The event has two attributes that record when the work began and when it should have been completed, allowing a computation of the portion of work done when this event occurs. The effect of the work to date is registered by calling the generic EN.REGISTER.EFFECT, which actually changes the terrain feature being worked on by this unit. The .COMPLETED.EFFECT is the portion completed and must be between 0 and 1.

ROUTINE EN.REGISTER.EFFECT
GIVEN .TASK, .FEATURE, .TECH, .SIDE, .SIZE, .X.LOC, .Y.LOC, .OC.PTR, .RESP.UNIT, .COMPLETED.EFFECT

When a job, done either explicitly by an engineer unit or implicitly by a nonengineer unit, has completed its final segment or has been interrupted at an intermediate segment with the potential that no more work will be done on it, the effect of the work to date is registered by this routine, i.e., the terrain feature that was the subject of the engineer activity is altered here. Proposed obstacles become active; obstacle breaches are created; defensive positions are completed; road capacities are upgraded; grid trafficability is improved for combat trails. The .COMPLETED.EFFECT is the portion completed and must be between 0 and 1.

ROUTINE EN.REQUEST.ENGINEER.PREPARATION
GIVEN .DP

This routine creates an engineer mission to prepare the given defensive position.

ROUTINE EN.REQUEST.ENGINEER.SUPPORT
GIVEN .UNIT, .TASK.TYPE

This routine responds to a decision table action by issuing a request for engineer support of the given .UNIT for a mission of the given .TASK.TYPE.

ROUTINE EN.REQUEST.OBS.COMP.MISSION
GIVEN .MISSION

The purpose of this routine is to decide whether the given obstacle complex mission should be done as one mission or split into several missions. If the obstacle complex data and engineer data offer the possibility of doing the job as one mission, nothing happens to the current mission. If the data divides the task of emplacing or breaching the obstacle complex into the component subtasks, this routine creates the separate missions and destroys the given aggregate mission.

ROUTINE EN.RESERVE.A.TECH.ARRAY
YIELDING .TECH.ARRAY

This routine builds a ragged array dimensioned by the number of tasks by the number of feature types. It is needed because the number of feature types for each type of task varies with each scenario. It will be used whenever needed to reserve a technique array.

ROUTINE EN.RETRIEVE.BRIDGING.ASSETS
GIVEN .SIDE, .FEATURE

This routine takes care of finding all retrievable assets owned by the given side at the site of the given bridge feature and returning them to their rightful owners. All retrievable assets at the given bridge site owned by the other side are destroyed.

ROUTINE EN.SEARCH.FOR.STAFF.AMONG.SUB
GIVEN .MVR, .MISSION, .FEATURE, .X.LOC, .Y.LOC, .SIZE

This routine searches down the requesting unit's command chain making a list of engineer staff units that are capable of doing the given mission. It does this by recursively looking at the capabilities of engineer support for subordinate maneuver units.

ROUTINE EN.SEND.WORK.CREW.HOME
GIVEN .WT

This routine breaks up a .WT owning equipment from more than one HQ unit by transferring all of the equipment owned by each other HQ unit to another work team and sending the newly created work team to the owning HQ's location. The given work team has no job at this point.

ROUTINE EN.SET.UNITS.COMBAT.STATUS
GIVEN .UNIT

This routine sets the combat status of the given engineer HQ unit.

ROUTINE EN.STOP.ALL.IMPLICIT.ENGINEER.WORK
GIVEN .UNIT

This routine causes implicit engineer task performance to end if the performing unit is moving to a point too far away from the work site for the job to continue. Partial completion of each task is computed from the technique being used and the start time attribute of the event.

ROUTINE EN.TRANSFER.ASSETS.HQ.TO.HQ
GIVEN .UNIT1, .UNIT2, .AMOUNT, .WPN.TYPE

This routine does all of the bookkeeping involved in transferring engineer assets from .UNIT1 to .UNIT2. .AMOUNT specifies the number of assets being transferred. .WPN.TYPE specifies the weapon type being transferred.

ROUTINE EN.TRANSFER.ASSETS.HQ.TO.WT
GIVEN .HQ, .WT, .AMOUNT, .TYPE

This routine does all of the bookkeeping involved in transferring engineer assets from the owning .HQ to .WT. .TYPE specifies the engineer asset prototype being transferred; .AMOUNT specifies the number of pieces of equipment. Only ..AVAILABLE assets at the .HQ can be transferred with this routine. Assets already assigned to another work team must be transferred with EN.TRANSFER.ASSETS.WT.TO.WT.

ROUTINE EN.TRANSFER.ASSETS.WT.TO.HQ
GIVEN .WT

This routine does the bookkeeping needed to transfer all of the engineer assets held by the .WT to the owning .HQ.

ROUTINE EN.TRANSFER.ASSETS.WT.TO.WT
GIVEN .WT1, .WT2, .AMOUNT, .TYPE

This routine does all of the bookkeeping involved in transferring .AMOUNT of engineer assets of .TYPE from one work team to another. It is used when equipment already assigned to work on one job is preempted by a job of higher priority.

ROUTINE EN.TRANSFER.JOB.SUPPLIES.TO.WT
GIVEN .JOB, .PHASE, .UNIT, .WT

This routine takes care of the bookkeeping necessary to transfer supplies required for the given .PHASE of the given .JOB from the given .UNIT to a work team.

ROUTINE EN.TRANSFER.SUPPLIES.FOR.EQUIPMENT
GIVEN .UNIT1, .UNIT2, .WG.PROTO, .WG.AMOUNT

This routine does all of the bookkeeping involved in transferring the fuel and ammunition requirements for equipment that has just been transferred from .UNIT1 to .UNIT2. .WG.PROTO is the weapon group type of the equipment, and .WG.AMOUNT is the change in the FL.STRENGTH upon which the supply amounts are based.

ROUTINE EN.TRANSFER.THIS.ASSET.WT.TO.WT
GIVEN .WT1, .WT2, .WT.ASSET

This routine does all of the bookkeeping involved in transferring a particular engineer asset, .WT.ASSET, from one work team .WT1 to another .WT2. This is done when assets need to be moved to a new work site while .WT1 stays at its present site and when .WT1 is breaking up and needs to send all of its assets to their owning HQ units.

ROUTINE EN.UPDATE.ALL.DEFENSIVE.PREPARATIONS.IN.PROGRESS

This routine updates GG.DEF.POSITION.STATUS, GG.UN.PORTION.OF.UNIT.EXPOSED and EN.DP.COMPLETION.FACTOR when a unit is at a path point where defensive preparation is taking place. The update occurs for all cases: explicit engineer preparation, implicit engineer preparation, and unit self-preparation using the unit prototype data.

ROUTINE EN.UPDATE.ARRAYS.AND.SEND.EXCESS.ON
GIVEN .WT

At the completion of a segment, the phase arrays are adjusted to reflect the numbers of equipment by type required for the completion of the phase. In addition, when a job's technique specifies that the organization point is at the work site, equipment may leave the work site as soon as its last use segment is completed without waiting for the completion of the phase. This routine locates the equipment that is no longer needed and routes it to the next job or to its owning HQ unit.

ROUTINE EN.UPDATE.ASSET.REPORT
GIVEN .ASSET

This routine prints a standard format report to the engineer history file so that an .ASSET's .STATUS can be traced. Because of the possible impact on run time when a large number of engineer assets are played, engineer input allows this report to be omitted for selected prototypes.

ROUTINE EN.UPDATE.DP.REPORT
GIVEN .DEF.POS, .STATUS

This routine prints a standard format report to the engineer history file so that a defensive position can be traced.

ROUTINE EN.UPDATE.JOB.REPORT
GIVEN .JOB, .STATUS, .INFO

This routine prints a standard format report to the engineer history file so that the progress of .JOB can be traced. .JOB is the job pointer, .STATUS is a defined-to-mean integer indicating how far the job has progressed, and .INFO is a real argument which allows information that is not a part of the job's attributes to be passed to this routine.

ROUTINE EN.WT.ARRIVAL.AT.DESTINATION
GIVEN .WT.GG

When an engineer work team arrives at its destination path point, three situations may exist: the unit is at its work site, the unit has returned to its .HQ after a job, the unit is rendezvousing with other work teams before proceeding to work. This routine analyzes the situation and initiates the appropriate action. The given argument .WT.GG is the GG.UNIT number of the work team.

ROUTINE EN.WT.DUMP
GIVEN .WT

This routine produces a work team status report for one of the interactive menu options of EN.PRINT.CONTROL.

7 ENGINEER MODULE CALLING TREE

Introduction

This chapter lists the calling sequences for the major processes of the EN module. Many EN module routines have multiple uses, and no attempt has been made to indicate all of the possible paths through the process. The sequences are presented in the way execution would normally proceed. Each sequence is entered through an EN event or from a routine or event external to the EN module. For any given routine in a sequence, its calling routine is the first routine preceding it and having three fewer .'s before its name.

Initialization

Before the simulation begins, VIC's initialization procedures read the input data and create the required entities and set structures. Processing of the input data involves some backtracking since the engineer mission data in GG, TB, and MF are input before any engineer information is available.

SS.LOAD.THE.DATABASE

This routine controls the loading of the input data for all modules chosen for inclusion in a run.

...EN.LOAD.AND.LIST.ENGINEER.DATA

This routine sequences the loading of the engineer data, which must follow the reading of input for system specification (SS), GG, GM, FL, TB, MF, and LO since each of those modules contains information crucial to the processing of the EN data.

......SS.POSITION.INPUT.FILE

This routine positions the input file reader after the module banner so that it is pointing to the first data item in the module.

......EN.LOAD.AND.LIST.DEFENSIVE.POSITION.DATA

This routine loads and lists the defensive position data. Then it establishes a defensive position prototype for each unit prototype and updates the position prototype of each defensive position created when the unit paths were read.

.........EN.UPDATE.DP.REPORT

This routine prints a standard format report to the engineer history file so that a defensive position can be traced.

.........EN.MISSION.DUMP

This routine produces a mission status report for one of the interactive menu options of EN.PRINT.CONTROL if the device is SS.THE.SCREEN and an output mission report if the device is SS.ENGINEER.HISTORY.FILE. It writes a report to the history file if any defensive position missions are canceled.

......EN.LOAD.JOB.PRIORITY.DATA

This routine loads engineer job prioritization data that allow the scenario builder to influence how a staff engineer unit will prioritize the jobs within its command chain.

......EN.LIST.JOB.PRIORITY.DATA

This routine lists engineer job prioritization data.

......EN.LOAD.ASSET.DATA

This routine loads the engineer asset prototype data. The names associated with the engineer asset prototypes have already been introduced as names of weapon prototypes.

.........FL.READ.NAME.OF.WPN

This utility routine reads a character string from the input file and associates with it the index of the corresponding weapon prototype.

......EN.LIST.ASSET.DATA

This routine lists the engineer asset prototype data to the echo file.

......EN.CREATE.HQ.UNITS

This routine builds the two lists of engineer HQ units in the database and establishes their equipment inventories and asset arrays. It also builds the lists of engineer staff units, which are HQ units with nonengineer GM.UN.SUPERIOR.

.........EN.UPDATE.ASSET.REPORT

This routine prints a standard format report to the engineer history file so that an asset's status can be traced. Because of the possible impact on run time when a large number of engineer assets are played, engineer input allows this report to be omitted for specific asset types.

......EN.LOAD.TECHNIQUE.DATA

This routine loads the engineer technique data, does the preprocessing of this data to allow quick searches, and builds the staff engineer asset and technique arrays.

.........SS.TRACE.AND.ABORT

This utility routine is called to stop the simulation with a message if an error occurs.

.........EN.READ.FEATURE.NAME

When technique data is input, this routine reads the feature name and associates with it the corresponding feature prototype number. This will vary by the task type of the technique, so several cases must be handled.

............SS.TRACE.AND.ABORT

This utility routine is called to stop the simulation with a message if an error occurs.

.........EN.INITIALIZE.TASK.TECH.ARRAY

This routine builds a ragged array dimensioned by side, task, feature, and technique number so it is known immediately which techniques are available for each engineer effort as determined by the first three dimensions.

.........EN.RESERVE.A.TECH.ARRAY

This routine builds a ragged array dimensioned by the number of tasks by the number of feature types. It is needed because the number of feature types for each type of task varies with each scenario. It will be used whenever we need to reserve a technique array.

.........EN.BUILD.STAFF.ARRAYS

This routine cycles through the techniques for each task and feature combination, comparing the total undamaged asset holdings of a staff and all of its engineer subordinates with the required equipment for the technique to determine whether or not this command chain can use the technique. This includes all techniques possible for this unit and its subordinates using any combination of equipment available to the group as a whole. Each element of the resulting technique array owned by the staff unit indicates the first technique for the corresponding task and feature combination the staff can use, or zero if it can use no technique.

............EN.ADD.AVAILABLE.ASSETS

This recursive routine cycles through a staff unit and all of its subordinates down the command chain, adding up all of the engineer equipment.

...............EN.ADD.AVAILABLE.ASSETS

............EN.COMPARE.OWNED.TO.REQUIRED

This routine compares owned assets with required assets and is used to decide which techniques a staff's command chain can use by comparing the total of its assets with the assets required for the technique.

............EN.DOES.THIS.COMMAND.HAVE.SUPPLIES

This recursive routine cycles through a staff unit and all of its subordinates looking for a unit whose supply inventory contains all of the supply types required to use the given technique.

...............EN.DOES.THIS.COMMAND.HAVE.SUPPLIES

......EN.LIST.TECHNIQUE.DATA

This routine lists the engineer technique data to the echo file.

.........GG.PRINT.UNIT.LEVEL

This routine associates a character string with each unit level index.

......EN.REQUEST.OBS.COMP.MISSION

The purpose of this routine is to decide whether the given obstacle complex mission should be done as one mission or be split into several missions. If the obstacle complex data and engineer data offer the possibility of doing the job as one mission, nothing happens to the current mission. If the data divides the task of emplacing or breaching the obstacle complex into the component subtasks, this routine creates the separate missions and destroys the given aggregate mission. All of the emplacement missions for obstacle complexes were generated when the MF data was read. Now that the technique data is available, each such mission must be examined to determine whether it should be divided into several missions or left as it is.

Mission Assignment

Every instance of a request for engineer effort becomes an engineer mission, and each side keeps a list of such missions. Periodically, the event EN.JOB.UPDATE causes a side to go through its missions

and attempt to assign them to HQ units. If such an assignment is possible, the mission is removed from the mission list and converted to a set of jobs. The jobs in turn are processed through an event EN.PROCESS.JOB.LIST.

SS.SCHEDULE.INITIAL.EVENTS

This routine takes care of scheduling the arrival of all deploying units at their first path point and of scheduling the first occurrence of all of the cyclic, self-scheduling events.

...EN.JOB.UPDATE

This event schedules itself after its initial scheduling by SS.SCHEDULE.INITIAL.EVENTS to periodically check the set of engineer missions for a side in an attempt to assign them to HQ units. The normal interval for this event is the job update interval from the EN input data. This event can be caused to be rescheduled for a shorter interval under certain situations (i.e., freeing up equipment, arrival of new supplies, or a change in combat status).

......EN.PROCESS.MISSION.LISTS

This routine periodically examines the mission list for each side, removing those missions which are assignable and creating jobs to be processed.

.........EN.ORDER.ENGINEER.WORK

This is the transition routine for moving an engineer mission to engineer jobs. If the work is impossible, the mission will be discarded. If the work involved in the mission is assignable, then this routine creates a job for each mission feature in the mission, sets the job attributes, and assigns the jobs to an engineer HQ unit. If the mission is unassignable because its earliest start time is still more than an update interval away or because no unit has the mission site in its current tactical area, then the mission remains on the mission list for the next cycle.

............EN.GET.TASK.SPECIFICS.FOR.MISSION

This routine is the link between the very specific tasks requested of engineers and the processing of those tasks in a generic fashion. The key yielded arguments are derived for each specific task type. Generic processing means that all jobs are handled by the same algorithms, using the yielded arguments from this routine to index into the technique array to determine how the job will be done and who will do it.

............EN.ASSIGN.ENGINEER.JOB

This routine handles the search for a unit on the given side to be responsible for the given mission. If the mission has a requesting unit, it first determines if the requesting unit is capable of doing the job. Then the search concentrates on finding engineer staff units whose command chain is capable of doing the work. If the mission has a requesting unit, that unit's direct engineer support is checked. Then the search continues down and up the chain of command. If no staff has been found or if the mission had no requesting unit, the routine looks for all staff units with a tactical area that contains the mission and whose command chain is capable. All qualifying staff unit's are kept in a set of possible units for the mission. If the mission can be assigned now, the staff with the smallest job load is chosen. Having determined a staff unit, the routine searches the staff unit's command chain for the unit at the command level specified by the chosen technique and closest to the job site.

...............EN.IS.THIS.NON.ENGINEER.CAPABLE

This routine is called to determine whether or not a nonengineer unit has the capability to perform the given engineer mission implicitly. It uses the technique data from the EN module, comparing the "engineer equipment" owned by the given unit with the equipment required for the techniques for this type of mission. If the equipment is owned, this routine returns the best technique, which is chosen on the basis of the criteria determined by the mission's given method (most preferred, shortest time, most effective).

..................EN.COMPARE.OWNED.TO.REQUIRED

This routine compares owned assets with required a s and is used to decide whether the owned assets are sufficient to meet the nee of the given mission.

..................EN.CHECK.FOR.SUPPLY.TYPES.IN.INVENTORY

This routine compares the supply list for a technique with the supply inventory for a unit and indicates whether or not the unit receives each type of required supply.

..................EN.CALCULATE.SEGMENT.DURATION

The purpose of this routine is to calculate the segment duration for a job of the given size using the given technique if the performing unit can use the amount of equipment indicated in the given array. The timing factors in the engineer input data are for segments using the preferred amount of required equipment. The duration must be adjusted upward as the amount of available equipment drops, with the required equipment minimum specifying the threshold at which the work cannot be done.

...............EN.ASSESS.DIRECT.ENGINEER.SUPPORT

This routine searches the nonengineer unit's subordinate list, making a list of its staff engineers capable of performing the given mission sometime during the course of the simulation.

..................EN.FIND.STAFF.ENGINEER

This routine links a GG.UNIT of type ..ENGINEER.UNIT to its associated staff unit, returning 0 if the unit is not a staff engineer.

..................EN.IS.THIS.STAFF.UNIT.CAPABLE

This routine cycles through the techniques for the given task and feature to find the best technique (determined by mission's search method: most preferred, fastest, best effect) for which the staff has the required equipment and receives the required supplies. If the staff is not able to do this type of work, the technique is zero.

.....................GG.COMPUTE.DISTANCE.TO.FEBA

This utility routine is used to determine the distance from the mission site to the FEBA. Each engineer technique has a minimum-distance-from-FEBA rule which must be satisfied if the technique is to be used for the mission. A unit's capability is checked without the FEBA rule, but a unit's assignability is dependent on it.

......................EN.COMPARE.OWNED.TO.REQUIRED

This routine compares owned assets with required assets and is used to decide whether the assets owned by a command chain are sufficient to meet the required equipment needs of a technique.

......................EN.DOES.THIS.COMMAND.HAVE.SUPPLIES

This recursive routine cycles through a staff unit and all of its HQ subordinates looking for a unit whose supply inventory contains all of the supply types required for the given technique.

.........................EN.CHECK.FOR.SUPPLY.TYPES.IN.INVENTORY

This routine compares the supply list for a technique with the supply inventory for an HQ unit and indicates whether or not the unit receives each type of required supply.

.........................EN.DOES.THIS.COMMAND.HAVE.SUPPLIES

......................EN.CALCULATE.SEGMENT.DURATION

The purpose of this routine is to calculate the segment duration for a job of the given size using the given technique if the performing unit can use the amount of equipment represented in the given equipment array. The timing factors given in the engineer input data are for segments using the preferred amount of required equipment. The duration must be adjusted upward as the amount of available equipment drops, with the required equipment minimum specifying the threshold at which the work cannot be done.

..................EN.IS.THIS.MISSION.INSIDE.TACTICAL.AREA

This routine determines if a unit has the given site in its current or future tactical area.

......................UT.IS.POINT.INSIDE.POLYGON

This utility routine decides whether a given point is inside or outside of a given polygon.

...............EN.SEARCH.FOR.STAFF.AMONG.SUB

This routine searches down the requesting unit's command chain making a list of engineer staff units that are capable of doing the job. It does this by recursively looking at engineer support for subordinate maneuver units.

..................EN.ASSESS.DIRECT.ENGINEER.SUPPORT

This routine searches the nonengineer unit's subordinate list, making a list of its staff engineers capable of performing the given mission some time during the course of the simulation.

..................EN.SEARCH.FOR.STAFF.AMONG.SUB

...............EN.FIND.ENGINEER.HQ

This routine links a GG.UNIT of type ..ENGINEER.UNIT with its associated HQ unit, returning 0 if the unit is not in the set of engineer HQ units.

................EN.FIND.STAFF.FOR.THIS.HQ

This routine cycles up the command chain from HQ until locating its staff unit.

...................EN.FIND.STAFF.ENGINEER

This routine links a GG.UNIT of type ..ENGINEER.UNIT to its associated staff unit, returning 0 if the unit is not a staff engineer.

...................SS.TRACE.AND.ABORT

This utility routine is called to stop the simulation with a message if an error occurs.

................EN.IS.THIS.STAFF.UNIT.CAPABLE

This routine cycles through the techniques for the given task and feature to find the best technique (determined by mission's search method: most preferred, fastest, best effect) for which the staff has the required equipment and receives the required supplies. If the staff is not able to do this type of work, the technique is zero.

................EN.IS.THIS.MISSION.INSIDE.TACTICAL.AREA

This routine determines if a unit has the given site in its current or future tactical area.

................EN.FIND.STAFF.ENGINEER

This routine links a GG.UNIT of type ..ENGINEER.UNIT to its associated staff unit, returning 0 if the unit is not a staff engineer.

................EN.DETERMINE.JOB.LOAD

This recursive routine cycles through a staff unit and all of its HQ subordinates and their HQ subordinates down the command chain, adding up all of the engineer jobs, both pending and active, to return the total number of jobs assigned to this command chain. It is used when more than one staff unit is being considered for a mission, the unit with the smallest job load being the prime candidate.

...................EN.FIND.ENGINEER.HQ

This routine links a GG.UNIT of type ..ENGINEER.UNIT with its associated HQ unit, returning 0 if the unit is not in the set of engineer HQ units.

...................EN.DETERMINE.JOB.LOAD

................EN.MOVE.DOWN.COMMAND.CHAIN

This routine moves one step down the command chain from a given engineer unit looking for the engineer HQ subordinate closest to the site, with the site in its tactical area and, if LO is played, with the required supply types in its inventory. It returns the GG.UNIT number of such a subordinate if it exists; otherwise, 0.

...................EN.FIND.ENGINEER.HQ

This routine links a GG.UNIT of type ..ENGINEER.UNIT with its associated HQ unit, returning 0 if the unit is not in the set of engineer HQ units.

..................EN.CHECK.FOR.SUPPLY.TYPES.IN.INVENTORY

This routine compares the supply list for a technique with the supply inventory for a unit and indicates whether or not the unit receives each type of required supply.

..................UT.IS.POINT.INSIDE.POLYGON

This utility routine decides whether a given point is inside or outside of a given polygon.

............EN.HOW.MUCH.TIME.IS.NEEDED

This routine totals the segment durations from a given segment to the last segment for the given technique for a job of the given size and returns the estimated remaining duration. This is an estimate of the fastest completion time possible and is used as a cut-off time for canceling jobs that cannot possibly be done in time.

............EN.DETERMINE.TRAIL.LOCATION

This routine finds the midpoint of the grid line closest to the unit responsible for building a trail through this grid. Once a mission to build a combat trail is assigned to an HQ unit, the location of the work site is chosen to be the middle of the side of the grid containing the trail and closest to the assigned HQ. This is to prevent having the work team suffer travel delays in trying to move through the grid to get to its work site before it can begin work.

............EN.UPDATE.JOB.REPORT

This routine prints a standard format report to the engineer history file so that the progress of a job can be traced.

.........EN.FIND.STAFF.FOR.THIS.HQ

This routine cycles up the command chain from HQ until its staff unit is located.

.........Schedule a EN.PROCESS.JOB.LIST

This event is scheduled whenever a staff needs to evaluate and process the jobs assigned to its command chain. Each pending job is then processed to see if equipment is available. This event is scheduled when equipment becomes available, supplies have been delivered, new jobs have been assigned, or the supported units have a new combat role. The time for the event to occur is one of the engineer timing input data items as determined by which of the trigger situations has occurred. In this particular sequence, the assignment of a new job triggers the responsible HQ unit to process its list.

.........EN.MISSION.DUMP

This routine produces a mission status report for one of the interactive menu options of EN.PRINT.CONTROL if the device is SS.THE.SCREEN and an output mission report if the device is SS.ENGINEER.HISTORY.FILE.

............EN.GET.TASK.NAME

This utility routine associates a text name with each task type being modeled.

......EN.CHECK.ON.WT.PROGRESS

This routine periodically checks every active work team to see if obstacle encounters will prevent the unit from getting to the job site in time to complete the job. If such a work team is found

and its equipment is not critical to the job, it is sent home and the job continues. If the job cannot be completed without that work team's equipment, the job is discontinued.

.........EN.DISCONTINUE.ENGINEER.MISSION

This routine does the bookkeeping necessary to cancel the mission of the work team and to send the work team back to its base. The work team is withdrawn from the phase of each job on its list, and an assessment is made as to whether each job can continue without the contribution of this work team.

......Schedule a EN.JOB.UPDATE

This is the self-scheduling of the update event.

Job Processing

The event which causes engineers to organize their assigned work and attempt to get it underway is EN.PROCESS.JOB.LIST. Several routines schedule or reschedule this event.

EN.PROCESS.MISSION.LISTS

This routine takes care of the periodic processing of a side's mission list. If new jobs are assigned, the event to process the job list for each responsible HQ's staff is rescheduled.

EN.PROCESS.THIS.MISSION.NOW

This routine takes care of dynamically-generated missions which must be processed immediately by trying to assign them to an engineer HQ unit and, if successful, scheduling the staff for the HQ unit to process its job list.

EN.END.OF.REST.PERIOD

This routine resets an asset's attributes after its rest period so that the asset may be assigned to jobs again. When an asset is resting, its priority is set so that no job can preempt the rest. The end of a rest period signals that previously unavailable equipment is ready for work.

EN.SEND.WORK.CREW.HOME

This routine breaks up a work team owning equipment from more than one HQ unit by transferring all of the equipment owned by each HQ unit to another work team and sending the newly created work team to the owning HQ's location. The original work team has no job at this point. The equipment that is returning to the HQ is now available for assignment to another job.

LO.AIR.CONVOY.ARRIVES.AT.UNIT

This event signals the arrival of new supplies by airlift. If the receiving unit is an engineer HQ unit, then that unit's staff must process its job list since some of its jobs may not have been started because of a lack of supplies.

LO.GET.SUPPLY.PROCESS

This process signals the arrival of new supplies by convoy. If the receiving unit is an engineer HQ unit, then that unit's staff must process its job list since some of its jobs may not have been started because of a lack of supplies.

LO.PREPLANNED.AIRDROP

This event signals the arrival of new supplies by air. If the receiving unit is an engineer HQ unit, then that unit's staff must process its job list since some of its jobs may not have been started because of a lack of supplies.

...EN.PROCESS.JOB.LIST

This event is scheduled whenever a staff needs to evaluate and process the jobs assigned to its command chain. The staff rebuilds its job list and prioritizes it. Each pending job is then processed in priority order to see if equipment is available. This event is scheduled when equipment becomes available, supplies have been delivered, new jobs have been assigned, or the supported unit has a new combat role. The time for the event to occur is one of the engineer timing input data items determined by which of the trigger situations has occurred.

......EN.PRIORITIZE.JOB.LIST

This routine prioritizes the entire list of engineer jobs for a staff engineer and its command chain.

.........EN.ADD.JOBS.TO.LIST

This recursive routine cycles through all of the units in the given staff's command chain adding a job to the staff unit's set of jobs for each active and pending job assigned to a subordinate. As each pending job is considered, if its latest completion time has passed, the job is canceled immediately.

............EN.FIND.ENGINEER.HQ

This routine links a GG.UNIT of type ..ENGINEER.UNIT with its associated HQ unit, returning 0 if the unit is not in the set of engineer HQ units.

............EN.UPDATE.JOB.REPORT

This routine prints a standard format report to the engineer history file so that the progress of a job can be traced.

............EN.ADD.JOBS.TO.LIST

.........EN.PRIORITIZE.INDIVIDUAL.JOB

This routine assigns a priority level to the given job. The priority is a real number derived from several contributing factors: the type of task linked with the combat status of the supported unit using the priority input array, the distance of the requestor from the task site if it is a mobility or survivability task, the margin of time between the earliest possible finish time and the latest acceptable completion time for the mission, the difference in level between the requestor and the responsible HQ, and whether or not the earliest start time of the mission has passed.

............EN.FIND.STAFF.FOR.THIS.HQ

This routine cycles up the command chain from HQ until locating its staff unit.

............EN.GET.TASK.MAPPING

This utility routine associates an engineer mission category with each task type being modeled (MOBILITY, COUNTERMOBILITY, SURVIVABILITY, GENERAL ENGINEERING).

......EN.PROCESS.JOB

This routine takes a job from the pending list, attempts to find the assets and supplies needed for its accomplishment, and, if found, creates the work units needed to bring the equipment together and move to the work site. If the given job becomes active, all other jobs of the same mission and at the same level of assignment are added to the work team lists in the order of their priority, and the work on them is done sequentially.

.........EN.FIND.STAFF.FOR.THIS.HQ

This routine cycles up the command chain from HQ until locating its staff unit.

.........EN.IS.THIS.STAFF.UNIT.CAPABLE

This routine cycles through the techniques for the given task and feature to find the best technique (determined by mission's search method: most preferred, fastest, best effect) for which the staff has the required equipment and receives the required supplies. If the staff is not able to do this type of work, the technique is zero. At this point, some time may have passed since the job was originally assigned to this HQ. The possibility exists that this staff's command chain can no longer do the job or that the chosen technique is no longer suitable for the situation.

.........EN.IS.THIS.MISSION.INSIDE.TACTICAL.AREA

This routine determines if a unit has the given site in its current or future tactical area.

.........EN.UPDATE.JOB.REPORT

This routine prints a standard format report to the engineer history file so that the progress of a job can be traced.

.........EN.HOW.MUCH.TIME.IS.NEEDED

This routine totals the segment durations from a given segment to the last segment for this technique for a job of the given size and returns the estimated remaining duration. This is an estimate of the fastest completion time possible and is used as a cut-off time for canceling jobs that cannot possibly be done in time.

.........EN.COMPUTE.DISTANCE.TO.JOB

This utility routine computes the distance between a given GG.UNIT and a given job.

.........EN.DISCONTINUE.ENGINEER.MISSION

This routine does the bookkeeping necessary to cancel the mission of the given work team and to send the work team back to its base. The work team is withdrawn from the phase of each job on its list, and an assessment is made as to whether each job can continue without the contribution of this work team. It is called if estimates of how long it will take to complete the job go beyond the latest completion time for the mission but work on earlier phases is in progress.

.........EN.DETERMINE.MINIMUM.EQUIPMENT.ARRAYS

This routine determines the minimum equipment required for a technique for the given phase of work and builds the minimum equipment array.

.........EN.ADD.AVAILABLE.ASSETS

This recursive routine cycles through a staff unit and all of its HQ subordinates and their HQ subordinates down the command chain, adding up the engineer equipment available at HQ units and those WT units working on lower priority jobs so that the given equipment array indicates all of the assets available within that command chain to a job of the given priority.

............EN.ADD.AVAILABLE.ASSETS

.........EN.COMPARE.OWNED.TO.REQUIRED

This routine compares owned assets with required assets and is used to decide whether the assets owned by a command chain and available or working on a lower priority job are sufficient to meet the needs of the current phase.

.........EN.CHECK.FOR.AVAILABILITY.OF.JOB.SUPPLIES

This routine compares the supply list for a technique with the supply inventory for a GG unit and indicates whether or not the unit has the required amount of each supply type on hand.

.........EN.DETERMINE.REQUIRED.EQUIPMENT.ARRAYS

This routine retrieves the required equipment information for the given phase of work using the given technique and builds the preferred equipment array.

.........EN.FORM.INITIAL.WT.UNITS

This routine searches a job's responsible unit's command chain for the equipment needed for the given phase. The equipment array indicates how much equipment is needed. The elements of this array approach zero as equipment is located. If all elements are zero, the desired equipment levels have been satisfied and the search stops. Prior processing assures that the minimum amount of the required equipment will be found. The search method finds as much of the needed equipment as is available.

............EN.BUILD.WORK.UNIT.AT.HQ

This recursive routine moves through an HQ and all HQ units subordinate to it searching for the engineer assets required for the given phase of the given job. The needed equipment array indicates what is still needed. As soon as the array is zero, the search stops. At each HQ, if needed equipment is found, it is transferred to a newly created work team, which will move it to the job. If LO is being played, each HQ gives its equipment the fuel and ammunition required.

...............EN.CREATE.A.WORK.UNIT

This routine retrieves an engineer work team from the free pool and sets its attributes.

...............EN.TRANSFER.ASSETS.HQ.TO.WT

This routine does all of the bookkeeping involved in transferring engineer assets from the owning HQ to a work team.

..................SS.TRACE.AND.ABORT

This utility routine is called to stop the simulation with a message if an error occurs.

..................EN.REFILE.ASSET.IN.INVENTORY

This routine takes care of removing an asset from its inventory list and refiling it at the bottom of the section of assets of its type.

..................EN.UPDATE.ASSET.REPORT

This routine prints a standard format report to the engineer history file so that an asset's status can be traced. Because of the possible impact on run time when a large number of engineer assets are played, engineer input allows this report to be omitted for specific asset types.

..................EN.TRANSFER.SUPPLIES.FOR.EQUIPMENT

This routine does all of the bookkeeping involved in transferring the fuel and ammunition requirements for equipment that has just been transferred from one unit to another.

..................FL.MOUNT.OR.DISMOUNT.A.UNIT

This routine is called when a unit's mounted or dismounted status has changed.

...............GG.MAP.FROM.GROUND.COMBAT.STATUS

This routine uses the ground combat status for a unit and the combat status mappings to set certain unit attributes.

...............EN.DETERMINE.WT.UNIT.PATH

This routine builds the path of a work team from its present location to its destination. This is a universal path builder, so code needs to be as generic as possible. Engineer work teams actually generate their paths dynamically, so some care must be taken to route them around known obstacles and to bridges for river crossings.

..................MF.REMOVE.UNIT.FROM.OBSTACLE.COMPLEXES

This routine is used to remove a work team from any obstacle complexes it currently occupies so that a new path may be determined for it.

..................EN.FIND.SH)RTEST.PATH.BETWEEN.TWO.POINTS

This routine recursively finds the shortest path from point A to point B.

.....................EN.FIND.OBSTACLE.BETWEEN.TWO.POINTS

This routine is a key part of the dynamic generation of work team paths. It finds the first obstacle which is either a line feature or a minefield located between given points A and B, and finds a point C by which the unit is able to bypass the obstacle.

........................EN.FIND.BYPASS.FOR.LINE.OBSTACLE

This routine finds the bypass point C for the given line obstacle.

........................EN.FIND.SHORTEST.PATH.BETWEEN.TWO.POINTS

..................FL.MOUNT.OR.DISMOUNT.A.UNIT

This routine is called when a unit's mounted or dismounted status has changed.

....GM.LOCATE.OBSTACLES.IN.UNIT.PATH

This is the lead routine for code-generated obstacle path points. It determines the location of all unit encounters with line features, barriers, minefields, and obstacle complexes and creates path points in the unit's path for each.

..................SS.CANCEL.EVENT.FOR.UNIT.IF.SCHEDULED

This routine cancels the event scheduled for a unit. VIC assumes that only one event is scheduled for a unit at any given time, and two unit attributes point to the event and the event type. Two cases are exceptions, the end of a line feature crossing and the end of a minefield delay. The unit event attributes are not set in either of these cases, and such events are not canceled by this routine but must be handled as special cases.

..................Schedule a GG.END.OF.WAIT

The scheduling of this event is the mechanism for starting the movement of an engineer work team. Work teams move in the same way that other GG.UNITs move. At the end of this sequence, work teams will not be controlled by the EN module again until they arrive at their destination.

..................EN.COMPUTE.WORK.TEAM.SPEED

This routine computes the speed at which a work team can travel, which is the minimum of the speeds of the assets it owns and the GG prototype speed for work teams.

.....................EN.FIND.ENGINEER.HQ

This routine links a GG.UNIT of type ..ENGINEER.UNIT with its associated HQ unit, returning 0 if the unit is not in the set of engineer HQ units.

..................EN.UPDATE.ASSET.REPORT

This routine prints a standard format report to the engineer history file so that an asset's status can be traced. Because of the possible impact on run time when a large number of engineer assets are played, engineer input allows this report to be omitted for specific asset types.

..................EN.FIND.ENGINEER.HQ

This routine links a GG.UNIT of type ..ENGINEER.UNIT with its associated HQ unit, returning 0 if the unit is not in the set of engineer HQ units.

...............EN.BUILD.WORK.UNIT.AT.HQ

............EN.BUILD.WORK.UNIT.AT.WT

This recursive routine moves down the command chain from an HQ, searching for equipment needed for the given phase of the given job held by work teams working on jobs of lower priority. If needed equipment is found, a new work team is created and the equipment is transferred to it to move to the new job site. If the preempted work team is currently using the equipment, the equipment leaves at the end of the current segment. The needed equipment array keeps track of what equipment is still being sought. The search continues until the array is 0 or the bottom of the chain is reached. Preprocessing

guarantees that the minimum amount of required equipment will be found, and the search method guarantees that as much of the preferred amount that is available will be sent.

...............EN.CREATE.A.WORK.UNIT

This routine retrieves an engineer work team from the free pool and sets its attributes.

...............EN.TRANSFER.ASSETS.WT.TO.WT

This routine does all of the bookkeeping involved in transferring an amount of engineer assets of a given type from one work team to another.

..................SS.TRACE.AND.ABORT

This utility routine is called to stop the simulation with a message if an error occurs.

..................EN.FILE.FREE.WEAPON.GROUP

This routine clears the attributes of a weapon group and files it in the free set to be used again when a work team needs a weapon group to hold assets being transferred to it.

..................EN.TRANSFER.SUPPLIES.FOR.EQUIPMENT

This routine does all of the bookkeeping involved in transferring the fuel and ammunition requirements for equipment that has just been transferred from one unit to another.

...............GG.MAP.FROM.GROUND.COMBAT.STATUS

This routine uses the ground combat status for a unit and the combat status mappings to set certain unit attributes.

...............EN.DETERMINE.WT.UNIT.PATH

This routine builds the path of a work team from its present location to its destination. This is a universal path builder, so code needs to be as generic as possible. Engineer work teams actually generate their paths dynamically, so care must be taken to route them around known obstacles and to bridges for river crossings.

...............EN.UPDATE.JOB.REPORT

This routine prints a standard format report to the engineer history file so that the progress of a job can be traced.

...............LO.MERGE.WITH.ANOTHER.UNIT

This routine takes care of transferring all of the supplies of one unit to another unit.

...............EN.DISSOLVE.WT.UNIT

When a work team has reached its final destination and has transferred all of its assets to another unit, this routine takes care of the bookkeeping involved in emptying all of the unit's sets and refiling it in the set of free units. Here, if the work team whose equipment was preempted now has neither assets nor jobs, it simply disappears.

...............EN.DISCONTINUE.ENGINEER.MISSION

This routine does the bookkeeping necessary to cancel the mission of the work team and to send the work team back to its base. The work team is withdrawn from the

phase of each job on its list, and an assessment is made as to whether each job can continue without the contribution of this work team. If the work team whose assets were preempted had jobs on its list but was in route to a job site, canceling the mission for the work team will take care of all the bookkeeping.

...............EN.FIND.ENGINEER.HQ

This routine links a GG.UNIT of type ..ENGINEER.UNIT with its associated HQ unit, returning 0 if the unit is not in the set of engineer HQ units.

...............EN.BUILD.WORK.UNIT.AT.WT

.........EN.TRANSFER.JOB.SUPPLIES.TO.WT

This routine transfers the required supplies for the mission to the first work team formed. That work team will move the supplies to the work site.

......Schedule a EN.PROCESS.JOB.LIST

This is the self-scheduling of the event for the given staff. A staff unit always has such an event scheduled after the first job is assigned to its command chain. The scheduling interval used here is the EN update interval unless the job list contains jobs that could not be started because the earliest start time had not arrived. Changes in the situation, as indicated by the routines at the top of this sequence, simply reschedule this event to occur earlier.

Work Team Movement

The event EN.PROCESS.JOB.LIST takes care of creating work teams and sending them on their way. The work teams move using the GG and GM movement processes and return to the EN module at a path point arrival at their destination.

GG.PATH.POINT.ARRIVAL

This is the key event in the ground unit movement process. Units move automatically from path point to path point. A path point arrival initiates a close look at the unit and what it should be doing. Several of the engineer sequences are entered through this event.

...EN.WT.ARRIVAL.AT.DESTINATION

When an engineer work team arrives at its destination path point, one of three situations exists: the unit is at its work site, the unit has returned to its HQ after a job, or the unit is rendezvousing with other work teams before proceeding to work. This routine analyzes the situation and initiates the appropriate action.

......EN.FIND.ENGINEER.HQ

This routine links a GG.UNIT of type ..ENGINEER.UNIT with its associated HQ unit, returning 0 if the unit is not in the set of engineer HQ units.

......EN.ARE.WTS.COLLOCATED

This routine determines whether two work teams are at the same location. Work teams formed to work on the same phase of the same mission always merge into one unit when they are at the same destination point.

......EN.TRANSFER.THIS.ASSET.WT.TO.WT

This routine does all of the bookkeeping involved in transferring a particular engineer asset from one work team to another. If the work team is collocated with another, the two merge equipment.

......LO.MERGE.WITH.ANOTHER.UNIT

This routine takes care of transferring all supplies owned by one unit to another unit. If the work team is merging with another, this routine moves the supplies.

......EN.DISSOLVE.WT.UNIT

When a work team has reached its final destination and has transferred all of its assets to another unit, this routine takes care of the bookkeeping involved in emptying all of the unit's sets and refiling it in the set of free units.

......EN.BEGIN.CURRENT.JOB.SEGMENT

If a work team arrives at its destination and it is the work site of the first job on its job list, then this routine checks to see if the next segment to be done on the job has been assigned to this unit. If that is the case and all of the equipment has arrived, the routine starts that segment. It is possible for work teams to arrive before earlier phases have been completed and for equipment to come from several locations to work on one segment. If the work team is assigned to a phase of the job that is not ready to begin, this routine does nothing; the completion of the preceding phase will trigger this team to begin work.

.........EN.CALCULATE.NEW.FINISH.TIME

This routine computes the duration of a segment of a job by comparing the assets of the work team doing the work with the equipment required for this segment of the given technique. This routine is only used when the work team has less equipment than the preferred amount for this technique. If the level of any of the required equipment is below the minimum specified by input data, then the duration is set high enough to cause the job to be discontinued. The asset array expresses in integer mode the number of pieces of equipment owned, while the EN.ASSET.UNDAMAGED.PORTION or the EN.WT.ASSET.PORTION of each asset expresses the usable portion of that asset. Work time is degraded if the equipment is not at 100 percent.

.........GM.SET.UNITS.WEATHER.TRAFFICABILITY.LEVEL

This routine determines the weather conditions at the work site to allow the calculation of job duration to include the impact of fair or poor weather.

.........SS.CANCEL.EVENT.FOR.UNIT.IF.SCHEDULED

This routine cancels the event scheduled for a unit. VIC assumes that only one event is scheduled for a unit at any given time, and two unit attributes point to the event and the event type. Since a new event is about to be scheduled for this unit, any other event already scheduled must be canceled. Actually, no other event for this unit should exist at this point.

.........Schedules a EN.END.OF.JOB.SEGMENT

This routine registers the completion of the current segment of the first job on a work team's job list and determines the work team's next course of action. The event is scheduled to occur after the passage of the calculated to be the segment duration given the present conditions at the work site.

.........EN.UPDATE.JOB.REPORT

This routine prints a standard format report to the engineer history file so that the progress of a job can be traced.

.........EN.UPDATE.ASSET.REPORT

This routine prints a standard format report to the engineer history file so that an asset's status can be traced. Because of the possible impact on run time when a large number of engineer assets are played, engineer input allows this report to be omitted for specific asset types.

.........EN.DISCONTINUE.ENGINEER.MISSION

This routine does the bookkeeping necessary to cancel the mission of the work team and to send the work team back to its base. The work team is withdrawn from the phase of each job on its list, and an assessment is made as to whether each job can continue without the contribution of this work team. Here, if the segment duration indicates that the job cannot be completed in time, the mission is canceled.

......GG.MAP.FROM.GROUND.COMBAT.STATUS

This routine uses the ground combat status for a unit and the combat status mappings to set certain unit attributes.

......EN.DETERMINE.WT.UNIT.PATH

This routine builds the path of a work team from its present location to its destination. This is a universal path builder, so code needs to be as generic as possible. Engineer work teams actually generate their paths dynamically, so some care must be taken to route them around known obstacles and to bridges for river crossings. In this case, if a work team was rendezvousing with others and all have arrived at the base location, this routine starts the resulting single work team on its way to the task site.

......EN.UPDATE.ASSET.REPORT

This routine prints a standard format report to the engineer history file so that an asset's status can be traced. Because of the possible impact on run time when a large number of engineer assets are played, engineer input allows this report to be omitted for specific asset types.

......EN.SEND.WORK.CREW.HOME

This routine breaks up a work team owning equipment from more than one HQ unit by transferring all of the equipment owned by each other HQ unit to another work team and sending the newly created work team to the owning HQ's location. The original work team has no job at this point. This routine is called when the work team has returned to the responsible HQ unit's base.

......SS.TRACE.AND.ABORT

This utility routine is called to stop the simulation with a message if an error occurs.

Job Performance

An engineer job moves through its successive segments by cycling between the routine EN.BEGIN.CURRENT.JOB.SEGMENT and the event EN.END.OF.JOB.SEGMENT.

EN.END.OF.JOB.SEGMENT

This event registers the completion of the current segment of the first job on a work team's job list and determines the work team's next course of action.

...EN.UPDATE.JOB.REPORT

This routine prints a standard format report to the engineer history file so that the progress of a job can be traced.

...LO.UPDATE.UNITS.SUPPLIES

This routine adds or subtracts a given amount of a given stock from the unit's supply inventory. Here it is called to consume the job supplies used in this segment.

...EN.UPDATE.ARRAYS.AND.SEND.EXCESS.ON

At the completion of a segment, the phase arrays are adjusted to reflect the numbers of equipment by type required for the completion of the phase. In addition, when a job's technique specifies that the organization point is at the work site, equipment may leave the work site as soon as its last use segment is completed without waiting for the completion of the phase. This routine locates the equipment that is no longer needed and routes it to the next job or to its owning HQ unit.

......EN.FILE.FREE.WEAPON.GROUP

This routine clears the attributes of a weapon group and files it in the free set to be used again when a work team needs a weapon group to hold assets being transferred to it.

......EN.UPDATE.ASSET.REPORT

This routine prints a standard format report to the engineer history file so that an asset's status can be traced. Because of the possible impact on run time when a large number of engineer assets are played, engineer input allows this report to be omitted for specific asset types.

......EN.CREATE.A.WORK.UNIT

This routine retrieves an engineer work team from the free pool and sets its attributes.

......EN.TRANSFER.THIS.ASSET.WT.TO.WT

This routine does all of the bookkeeping involved in transferring a particular engineer asset from one work team to another.

......GG.MAP.FROM.GROUND.COMBAT.STATUS

This routine uses the ground combat status for a unit and the combat status mappings to set certain unit attributes.

......EN.DETERMINE.WT.UNIT.PATH

This routine builds the path of a work team from its present location to its destination. This is a universal path builder, so code needs to be as generic as possible. Engineer work teams actually generate their paths dynamically, so some care must be taken to route them around known obstacles and to bridges for river crossings.

......EN.SEND.WORK.CREW.HOME

This routine breaks up a work team owning equipment from more than one HQ unit by transferring all of the equipment owned by each other HQ unit to another work team and sending

the newly created work team to the owning HQ's location. The original work team has no job at this point.

...EN.HOW.MUCH.TIME.IS.NEEDED

This routine totals the segment durations from a given segment to the last segment for this technique for a job of the given size and returns the estimated remaining duration. This is an estimate of the fastest completion time possible and is used as a cut-off time for canceling jobs that cannot possibly be done in time. Here it is called to update the job's estimated duration.

...EN.BEGIN.CURRENT.JOB.SEGMENT

With the completion of a segment of a job, this routine checks to see if the next segment to be done on the job has been assigned to the current unit and starts that segment if that is the case and all of the equipment has arrived. It is possible for work teams to arrive before earlier phases have been completed and for equipment to come from several locations to work on one segment. If the work team is assigned to a phase of the job that is not ready to begin, this routine does nothing; the completion of the last segment of the preceding phase will trigger this team to begin work.

...EN.CLOSE.OUT.JOB.PHASE

This routine does the bookkeeping that is needed when a work team completes the last segment of a phase of a job. The job attributes must be updated, and the work team sent to either its next job or home if it is finished.

......EN.UPDATE.JOB.REPORT

This routine prints a standard format report to the engineer history file so that the progress of a job can be traced.

......EN.REGISTER.EFFECT.OF.JOB

When a job being performed by an engineer unit has reached and completed its final segment, has completed a segment with end effect of 1, or has been interrupted at an intermediate segment with the potential that no more work will be done on it, this routine extracts the necessary parameters for a call to the generic EN.REGISTER.EFFECT used for implicit as well as explicit engineer activity.

.........EN.REGISTER.EFFECT

When a job, done either explicitly by an engineer unit or implicitly by a nonengineer unit, has completed its final segment, has completed a segment with end effect of 1, or has been interrupted at an intermediate segment with the potential that no more work will be done on it, the effect of the work to date is registered by this routine. This routine considers each type of task and takes the appropriate action to ensure that the terrain feature that has been altered by the work appears properly.

............MF.REMOVE.MINEFIELD

This routine makes a minefield defunct if breaching or clearing operations have left it ineffective. The minefield is removed from the set of minefields but is not destroyed as an entity to avoid the "reference to destroyed entity" error.

............EN.CREATE.A.MISSION.REQUEST

This routine creates an engineer mission for the given data. The assumption is that this is a dynamically generated new mission involving only one feature and that it should be

processed immediately. A check is made for duplicate work first, and no new mission is created if the same work is already in the system. In this case, completion of certain types of engineer work generates new mission. For example, completion of a bridge using retrievable assets generates a mission to improve the bridge and retrieve the assets.

............TB.LOCATE.FS.INTERSECTION

This utility routine determines the point at which a path segment crosses a line feature. It is needed because unit encounters with line features keep track of where a unit's center is when the leading edge strikes the obstacle. The location of the leading edge must be calculated.

............EN.RETRIEVE.BRIDGING.ASSETS

This routine takes care of finding all retrievable assets owned by engineers at the site of the given bridge feature and returning them to their rightful owners.

...............EN.CREATE.A.WORK.UNIT

This routine retrieves an engineer work team from the free pool and sets its attributes.

...............EN.UPDATE.ASSET.REPORT

This routine prints a standard format report to the engineer history file so that an asset's status can be traced. Because of the possible impact on run time when a large number of engineer assets are played, engineer input allows this report to be omitted for specific asset types.

...............EN.SEND.WORK.CREW.HOME

This routine breaks up a work team owning equipment from more than one HQ unit by transferring all of the equipment owned by each other HQ unit to another work team and sending the newly created work team to the owning HQ's location. The original work team has no job at this point.

............EN.UPDATE.DP.REPORT

This routine prints a standard format report to the engineer history file so that a defensive position can be traced.

......EN.GET.TASK.NAME

This utility routine associates a text name with each task type being modeled.

......SS.TRACE.AND.ABORT

This utility routine is called to stop the simulation with a message if an error occurs.

......GG.MAP.FROM.GROUND.COMBAT.STATUS

This routine uses the ground combat status for a unit and the combat status mappings to set certain unit attributes.

......EN.DETERMINE.WT.UNIT.PATH

This routine builds the path of a work team from its present location to its destination. This is a universal path builder, so code needs to be as generic as possible. Engineer work teams actually generate their paths dynamically, so some care must be taken to route them around known obstacles and to bridges for river crossings.

......EN.SEND.WORK.CREW.HOME

This routine breaks up a work team owning equipment from more than one HQ unit by transferring all of the equipment owned by each other HQ unit to another work team and sending the newly created work team to the owning HQ's location. The original work team has no job at this point.

...EN.FIND.STAFF.FOR.THIS.HQ

This routine cycles up the command chain from HQ until locating its staff unit.

...EN.PROCESS.JOB

This routine takes a job from the pending list, attempts to find the assets and supplies needed for its accomplishment, and, if found, creates the work units needed to bring the equipment together and move to the work site. If the given job becomes active, all other jobs of the same mission and at the same level of assignment are added to the work team lists in the order of their priority, and the work on them is done sequentially. In this case, a job at the end of a phase with no assignments made for later phases is put back on the pending list and must be processed again.

...EN.IS.WT.AT.WORK.SITE

This utility routine determines whether the given work team is at the site of its first job. In this case, a search is made of work teams assigned to later phases of the job to determine if any are at the work site. If so, another call is made to EN.BEGIN.CURRENT.JOB.SEGMENT to have the work proceed.

Dynamic Generation of New Engineer Missions

Engineer missions are generated in two ways. The obstacle plan from the TB and MF input data generate all the obstacle emplacement missions except for those arising from the external event EN.BRIDGE.DEMOLITION. The path point preparation data from the GG input data generates all of the survivability missions. Otherwise, missions arise from the occurrence of certain situations. Most of these missions follow the first sequence given below. The exceptions are minefield and obstacle complex breaches. These missions are treated differently because of the scaling of effect on the encountering unit. The second sequence given below indicates how these two special cases are handled.

LO.UPDATE.CONVOY.PASSAGE

This routine updates the attributes of a link in the road network after the passage of a convoy. The possibility exists that the road has reached its threshold for requiring maintenance. If so, an engineer mission to maintain the road is generated.

LO.UPDATE.NODE.OCCUPATION

This routine periodically checks each link in the road network for ownership and usability. If the trafficability of the link makes it unusable, a mission to maintain the road or improve a damaged bridge is generated.

GG.UPDATE.RESERVE.ON.ROAD

This routine updates the attributes of a link in the road network after the passage of a maneuver unit. The possibility exists that the road has reached its threshold for requiring maintenance. If so, an engineer mission to maintain the road is generated.

TB.CROSSING.FEATURE

This routine registers a unit's encounter with a line obstacle. If no breaches exist for the unit to use, an engineer mission to create a breach is generated.

MF.UNIT.HITS.MINEFIELD

This routine registers a unit's encounter with a minefield. If it is an enemy minefield, an engineer mission to clear the minefield is generated. Note that breaching of the minefield is handled in a different process because of the scaling of the effect on the unit. Even though the clearing mission is automatically generated, the user controls the extent to which clearing operations are processed by appropriate engineer data. For example, the mission type is totally disabled if the engineer data contains no technique for the task.

EN.BRIDGE.DEMOLITION

This external event generates a mission to prepare a bridge for demolition. The user has several mechanisms to control the tactical destruction of bridges.

EN.CREATE.MISSION.FOR.A.PLANNING.POINT

This routine is the action part of the engineer planning process. Each maneuver unit's path is scanned for known obstacles, and planning path points are created to allow the unit to request mobility tasks in advance of the actual encounter. An arrival at the planning path point signals the call to this routine to generate the mission.

EN.REGISTER.EFFECT

This routine registers the change in the terrain feature that was the subject of engineer activity. It may also be responsible for generating new engineer missions.

...EN.CREATE.A.MISSION.REQUEST

This routine creates an engineer mission for the given data. The assumption is that this is a dynamically generated new mission involving only one feature and that it should be processed immediately. A check is made for duplicate work first, and no new mission is created if the same work is already in the system.

......EN.PROCESS.THIS.MISSION.NOW

This routine takes care of dynamically-generated missions which must be processed immediately. An attempt is made to assign the mission to an engineer HQ unit, and, if successful, that HQ's staff is scheduled to process the job list.

.........EN.ORDER.ENGINEER.WORK

This is the transition routine for moving an engineer mission to engineer jobs. Yielded arguments indicate whether the work is assignable, unassignable, or impossible. If the work involved in the mission is assignable, then this routine creates a job for each mission feature in the mission, sets the job attributes, and assigns the jobs to an engineer HQ unit.

.........EN.FIND.STAFF.FOR.THIS.HQ

This routine cycles up the command chain from HQ until locating its staff unit.

.........Schedule a EN.PROCESS.JOB.LIST

This event is scheduled whenever a staff needs to evaluate and process the jobs assigned to its command chain. The staff rebuilds its job list and prioritizes it. Each pending job is then

processed to see if equipment is available. This event is scheduled when equipment becomes available, supplies have been delivered, new jobs have been assigned, or the supported units have a new combat role. The time for the event to occur is one of the engineer timing input data items as determined by which of the trigger situations has occurred.

.........EN.PERFORM.IMPLICIT.TASKS

This routine allows the given maneuver unit to perform missions implicitly on its side's mission list. Such missions must have this unit as their requestor, and the unit must own the equipment required in the engineer technique data for doing the type of work indicated by the mission's task and feature type. Prior processing has determined the unit's capability and filed the unit in the set of possible units for the mission. This routine takes care of finding those missions which are presently within the unit's radius of control, determining whether equipment is available, and scheduling work completion if it is.

............EN.GET.TASK.MAPPING

This utility routine associates an engineer mission category with each task type being modeled (MOBILITY, COUNTERMOBILITY, SURVIVABILITY, GENERAL ENGINEERING).

............WT.CALCULATE.WEATHER.XY.INDICES

This utility routine determines the weather indices for the given location so that the current weather can be determined.

............EN.GET.TASK.SPECIFICS.FOR.MISSION

This routine is the link between the very specific tasks requested of engineers and the processing of those tasks in a generic fashion. The key yielded arguments are derived for each specific task. Generic processing means that all jobs are handled by the same algorithms, using the above arguments to index into the technique array to determine how and by whom the job will be done.

............EN.COMPARE.OWNED.TO.REQUIRED

This routine compares owned assets with required assets and is used when the assets owned by the unit are sufficient to meet the needs of the technique chosen for the work.

............EN.CHECK.FOR.AVAILABILITY.OF.JOB.SUPPLIES

This routine compares the supply list for a technique with the supply inventory for a GG unit and indicates whether or not the unit has the required amount of each type of supply on hand.

............EN.CALCULATE.SEGMENT.DURATION

This routine calculates the segment duration for a job of the given size using the given technique if the performing unit can use the amount of equipment indicated in the given array. The timing factors given in the engineer input data are for segments using the preferred amount of required equipment. The duration must be adjusted upward as the amount of available equipment drops, with the required equipment minimum specifying the threshold at which the work cannot be done.

............LO.UPDATE.UNITS.SUPPLIES

This routine adds or subtracts a given amount of a given stock from the unit's supply inventory. In this case, if the unit is performing an engineer task implicitly, the supplies for the work are consumed at the beginning of the work.

............Schedules a EN.REGISTER.EFFECT.OF.NON.ENGINEER.JOB

When a job being done implicitly by a nonengineer unit has reached and completed its final segment or has been interrupted at an intermediate segment with the potential that no more work will be done on it, this routine extracts the necessary parameters for a call to the generic routine EN.REGISTER.EFFECT.

......EN.REQUEST.OBS.COMP.MISSION

The purpose of this routine is to decide whether the given obstacle complex mission should be done as one mission or split into several missions. If the obstacle complex data and engineer data offer the possibility of doing the job as one mission, nothing happens to the current mission here. If the data divides the task of emplacing or breaching the obstacle complex into the component subtasks, this routine creates the separate missions and destroys the given aggregate mission. In this particular sequence, a request to breach an obstacle complex from a planning point has been made. Unlike a request that arises from an actual encounter, this type of mission requires that all of the complex must be breached.

.........EN.PROCESS.THIS.MISSION.NOW

This routine takes care of dynamically-generated missions which must be processed immediately. For obstacle complex missions, the decision whether to work on the obstacle as a whole or in parts must precede any attempt to process the mission(s).

Because of the scaling of delay for units encountering minefields and obstacle complexes, special calculations must be made to determine whether engineer capability can help the unit move through the obstacle faster than it can without help.

Engineers can alter only the breaching delay assessed to a unit that is not using a bypass tactic. The unscaled breaching delay is computed first and is compared with the engineer breaching duration. If the engineer time is faster, then the engineer mission is generated and the unit's breaching delay is reduced to the engineer job duration. The discovery delay and the crossing delay are added to the breaching delay, and the total delay is then multiplied by the fraction of unit actually hitting the minefield. The result of this calculation becomes the assessed delay to the unit.

With obstacle complexes, the calculation and generation of engineer missions is the same except for one additional step. We first compute what portion of the total complex is seen by the unit. Then that portion is multiplied by the number of unbreached obstacles of a given type and rounded to the nearest integer to determine how many obstacles of that type must be breached. For example, if a complex contains five unbreached minefields of a certain type and the unit sees half of the complex, then the unit must breach three of the minefields.

MF.UNIT.HITS.MINEFIELD

This routine is called when a unit encounters a minefield to assess appropriate delays and discovery attrition. It schedules the end of the minefield delay for the unit; the unit is stopped at the minefield entrance for the assessed delay. Crossing attrition is assessed at the end of the delay.

MF.ENTER.OBSTACLE.COMPLEX

This routine is called when a unit enters an obstacle complex. It determines what portion of the complex is actually seen by the unit and calculates appropriate delays. The delay is spread along the unit's path through the complex in a way that allows the unit to continue moving but at a slower speed. Crossing attrition is assessed when the unit exits the complex.

...EN.ASSESS.ENGINEER.CAPABILITY

This routine determines whether engineers currently have the capability to breach the given number of obstacles in less time than the unit's computed delay if the breaches are done without engineer help. If engineer work is already in progress, the job duration is returned as the difference of the expected completion time and the current time. If no work is already in progress, both explicit and implicit capabilities are checked. If either work effort will produce a faster breach, the fastest method is chosen. In this case, the unit is assessed the shorter delay and this routine generates the actual engineer work.

......TB.FIND.GRID.SQUARE.FOR.POINT

This utility routine returns a grid pointer for the grid containing a given point.

......EN.FIND.ENGINEER.HQ

This routine links a GG.UNIT of type ..ENGINEER.UNIT with its associated HQ unit, returning 0 if the unit is not in the set of engineer HQ units.

......EN.ASSIGN.ENGINEER.JOB

This routine handles the search for a unit on the given side to be responsible for the given mission. If the mission has a requesting unit, it first determines if the requesting unit is capable of doing the job. Then the search concentrates on finding engineer staff units whose command chain is capable of doing the work. If the mission has a requesting unit, that unit's direct engineer support is checked. The search continues down the chain of command and then moves up the chain. If no staff has been found or if the mission had no requesting unit, the routine looks for a staff unit with a tactical area that contains the mission and whose command chain is capable. Having determined a staff unit, the routine searches the staff unit's command chain for the unit at the command level specified by the chosen technique and closest to the job site.

......WT.CALCULATE.WEATHER.XY.INDICES

This utility routine determines the weather indices for the given location so that the current weather can be determined.

......EN.COMPARE.OWNED.TO.REQUIRED

This routine compares owned assets with required assets and is used to decide whether the assets owned by a unit and available are sufficient to meet the needs of the chosen technique.

......EN.CHECK.FOR.AVAILABILITY.OF.JOB.SUPPLIES

This routine compares the supply list for a technique with the supply inventory for a GG unit and indicates whether or not the unit has the required amount of each type of supply on hand.

......EN.CALCULATE.SEGMENT.DURATION

This routine calculates the segment duration for a job of the given size using the given technique if the performing unit can use the amount of equipment indicated in the given array. The timing factors given in the engineer input data are for segments using the preferred amount of required

equipment. The duration must be adjusted upward as the amount of available equipment drops, with the required equipment minimum specifying the threshold at which the work cannot be done.

......EN.PROCESS.THIS.MISSION.NOW

This routine takes care of dynamically-generated missions which must be processed immediately.

......LO.UPDATE.UNITS.SUPPLIES

This routine adds or subtracts a given amount of a given stock from the unit's supply inventory.

......Schedules a EN.REGISTER.EFFECT.OF.NON.ENGINEER.JOB

If the processing here generates implicit engineer work, then the completion of the work is scheduled.

Engineer Asset Attrition

In any routine which alters an engineer unit's weapon group strength, a call must be made to the routine that reconciles the engineer records with the weapon groups The following routines alter a unit's weapon group strength:

GG.CHANGE.MASS.OF.UNIT
GG.CHECK.FOR.BREAK
FL.UPDATE.MASS.OF.UNIT
AG.AREA.DAMAGE
AG.POINT.DAMAGE
AT.AREA.DAMAGE
AT.POINT.DAMAGE
CH.CHANGE.MASS.OF.UNIT
MF.ACCOUNT.FOR.MINEFIELD.DISCOVERY.LOSSES
MF.ACCOUNT.FOR.MINEFIELD.CROSSING.LOSSES
RD.DETERMINE.RAM.FAILURES
RD.ALLOCATE.WPN.REPAIRS

...EN.RECONCILE.ASSET.COUNTS

This routine balances an engineer unit's weapons set with its engineer inventory when changes occur because of attrition or return-to-duty repairs. Engineer assets are also weapons, and the FL module sees them only as weapons when it computes attrition. In order to bring the engineer bookkeeping in line with the weapon counts, this routine must compare totals and adjust accordingly.

......EN.FIND.ENGINEER.HQ

This routine links a GG.UNIT of type ..ENGINEER.UNIT with its associated HQ unit, returning 0 if the unit is not in the set of engineer HQ units.

......EN.FIND.STAFF.FOR.THIS.HQ

This routine cycles up the command chain from HQ until locating its staff unit.

......EN.REFILE.ASSET.IN.INVENTORY

This routine takes care of removing an asset from its inventory list and refiling it at the bottom of the section of assets of its type. This action would be taken in this sequence if an asset is declared to be damaged (unusable).

......EN.UPDATE.ASSET.REPORT

This routine prints a standard format report to the engineer history file so that an asset's status can be traced. Because of the possible impact on run time when a large number of engineer assets are played, engineer input allows this report to be omitted for specific asset types.

......RD.PLACE.LOSSES.IN.RECOVERY.SYSTEM

This routine is the entry point to the return-to-duty process for damaged weapons. With engineer assets, care must be taken to ensure that the return-to-duty records indicate that the owning HQ unit is to receive repaired weapons and not the work team that was temporarily using them.

......LO.UPDATE.UNIT.SUPPLIES.DUE.TO.LOSS

This routine adjusts supply levels to indicate a loss of supplies equal in proportion to the loss of the corresponding weapons.

......EN.FILE.FREE.WEAPON.GROUP

This routine clears the attributes of a weapon group and files it in the free set to be used again when a work team needs a weapon group to hold assets being transferred to it.

......EN.UPDATE.JOB.REPORT

This routine prints a standard format report to the engineer history file so that the progress of a job can be traced.

......EN.DISCONTINUE.ENGINEER.MISSION

This routine does the bookkeeping necessary to cancel the mission of the work team and to send the work team back to its base. The work team is withdrawn from the phase of each job on its list, and an assessment is made as to whether each job can continue without the contribution of this work team. In this sequence, if all of a work team's assets are destroyed, then its mission is discontinued.

.........EN.FIND.ENGINEER.HQ

This routine links a GG.UNIT of type ..ENGINEER.UNIT with its associated HQ unit, returning 0 if the unit is not in the set of engineer HQ units.

.........EN.IS.WT.CRUCIAL.TO.PHASE.COMPLETION

This routine determines whether or not the phase of a mission that the given work team is assigned can be completed without it. It indicates whether or not the work team has equipment that will cause the available equipment levels to fall below the minimum thresholds.

.........SS.CANCEL.EVENT.FOR.UNIT.IF.SCHEDULED

This routine cancels the event scheduled for a unit. VIC assumes that only one event is scheduled for a unit at any given time, and two unit attributes point to the event and the event type. Two cases are exceptions, the end of a line feature crossing and the end of a

minefield delay. The unit event attributes are not set in either of these cases, and such events are not canceled by this routine but must be handled as special cases.

.........EN.SEND.WORK.CREW.HOME

This routine breaks up a work team owning equipment from more than one HQ unit by transferring all of the equipment owned by each other HQ unit to another work team and sending the newly created work team to the owning HQ's location. The original work team has no job at this point.

............EN.CREATE.A.WORK.UNIT

This routine retrieves an engineer work team from the free pool and sets its attributes.

...............GG.COMPUTE.DISTANCE.TO.FEBA

This utility routine computes a unit's current distance from the FEBA and is used to set unit's attribute.

...............TR.SET.TERRAIN.ATTRIBUTES.OF.UNIT

This utility routine sets the appropriate unit terrain attributes according to the unit's present location.

............EN.TRANSFER.THIS.ASSET.WT.TO.WT

This routine does all of the bookkeeping involved in transferring a particular engineer asset from one work team to another.

...............SS.TRACE.AND.ABORT

This utility routine is called to stop the simulation with a message if an error occurs.

...............EN.FILE.FREE.WEAPON.GROUP

This routine clears the attributes of a weapon group and files it in the free set to be used again when a work team needs a weapon group to hold assets being transferred to it.

...............EN.TRANSFER.SUPPLIES.FOR.EQUIPMENT

This routine does all of the bookkeeping involved in transferring the fuel and ammunition requirements for equipment that has just been transferred from one unit to another.

..................LO.FIND.STOCK.POINTER

This utility routine looks through a unit's inventory and returns the pointer for the stock of a given type.

..................LO.UPDATE.UNITS.SUPPLIES

This routine adds or subtracts a given amount of a given stock from the unit's supply inventory.

............GG.MAP.FROM.GROUND.COMBAT.STATUS

This routine uses the ground combat status for a unit and the combat status mappings to set certain unit attributes.

............EN.DETERMINE.WT.UNIT.PATH

This routine builds the path of a work team from its present location to its destination. This is a universal path builder, so code needs to be as generic as possible. Engineer work teams actually generate their paths dynamically, so care must be taken to route them around known obstacles and to bridges for river crossings.

............EN.DISSOLVE.WT.UNIT

When a work team has reached its final destination and has transferred all of its assets to another unit, this routine takes care of the bookkeeping involved in emptying all of the unit's sets and refiling it in the set of free units.

...............SS.CANCEL.EVENT.FOR.UNIT.IF.SCHEDULED

This routine cancels the event scheduled for a unit. VIC assumes that only one event is scheduled for a unit at any given time, and two unit attributes point to the event and the event type. Two cases are exceptions, the end of a line feature crossing and the end of a minefield delay. The unit event attributes are not set in either of these cases, and such events are not canceled by this routine but must be handled as special cases.

...............SS.TRACE.AND.ABORT

This utility routine is called to stop the simulation with a message if an error occurs.

...............EN.FILE.FREE.UNIT.IN.POOL

This routine removes an engineer work team from active status and files it in the free pool.

..................GG.DELETE.THE.PATH

..................TB.SET.TERRAIN.ATTRIBUTES.OF.UNIT

..................SS.TRACE.AND.ABORT

This utility routine is called to stop the simulation with a message if an error occurs.

............EN.IS.WT.AT.BASE

This utility routine determines whether the given work team is at its base HQ location.

............EN.TRANSFER.ASSETS.WT.TO.HQ

This routine does the bookkeeping needed to transfer all of the engineer assets held by a work unit to the owning HQ.

...............Schedules an EN.END.OF.REST.PERIOD

This routine resets an asset's attributes after its rest period so that the asset may be assigned to jobs again. When an asset is resting, its priority is set so that no job can preempt the rest period.

...............EN.UPDATE.ASSET.REPORT

This routine prints a standard format report to the engineer history file so that an asset's status can be traced. Because of the possible impact on run time when a large number of engineer assets are played, engineer input allows this report to be omitted for specific asset types.

...............SS.TRACE.AND.ABORT

This utility routine is called to stop the simulation with a message if an error occurs.

...............LO.MERGE.WITH.ANOTHER.UNIT

...............EN.FILE.FREE.WEAPON.GROUP

This routine clears the attributes of a weapon group and files it in the free set to be used again when a work team needs a weapon group to hold assets being transferred to it.

...............FL.MOUNT.OR.DISMOUNT.A.UNIT

This routine is called when a unit's mounted or dismounted status has changed.

............EN.FIND.STAFF.FOR.THIS.HQ

This routine cycles up the command chain from HQ until locating its staff unit.

.........GG.MAP.FROM.GROUND.COMBAT.STATUS

This routine uses the ground combat status for a unit and the combat status mappings to set certain unit attributes.

.........EN.DETERMINE.WT.UNIT.PATH

This routine builds the path of a work team from its present location to its destination. This is a universal path builder, so code needs to be as generic as possible. Engineer work teams actually generate their paths dynamically, so care must be taken to route them around known obstacles and to bridges for river crossings.

.........EN.DISSOLVE.WT.UNIT

When a work team has reached its final destination and has transferred all of its assets to another unit, this routine takes care of the bookkeeping involved in emptying all of the unit's sets and refiling it in the set of free units.

.........EN.FIND.STAFF.FOR.THIS.HQ

This routine cycles up the command chain from HQ until locating its staff unit.

.........EN.REGISTER.EFFECT.OF.JOB

When a job being performed by an engineer unit has reached and completed its final segment or has been interrupted at an intermediate segment with the potential that no more work will be done on it, this routine extracts the necessary parameters for a call to the generic EN.REGISTER.EFFECT used for implicit as well as explicit engineer activity.

.........EN.UPDATE.JOB.REPORT

This routine prints a standard format report to the engineer history file so that the progress of a job can be traced.

......EN.CALCULATE.NEW.FINISH.TIME

This routine computes the duration of a segment of a job by comparing the assets of the work team doing the work with the equipment required for this segment of the given technique. This routine is used only when the work team has less equipment than the preferred amount for this technique. If the level of any of the required equipment is below the minimum specified by

input data, then the duration is set high enough to cause the job to be discontinued. The asset array expresses in integer mode the number of pieces of equipment owned, while the EN.ASSET.UNDAMAGED.PORTION or the EN.WT.ASSET.PORTION of each asset expresses the usable portion of that asset. Work time is degraded if the equipment is not at 100 percent. In this sequence, if a work team has suffered damage and is presently working, then the impact of the damage must be registered by extending the work time. The possibility exists that the work cannot continue with the present equipment levels.

.........EN.DISCONTINUE.ENGINEER.MISSION

This routine does the bookkeeping necessary to cancel the mission of the work team and to send the work team back to its base. The work team is withdrawn from the phase of each job on its list, and an assessment is made as to whether each job can continue without the contribution of this work team. In this sequence, if equipment levels are now below the minimum required, then the mission is discontinued.

......SS.TRACE.AND.ABORT

This utility routine is called to stop the simulation with a message if an error occurs.

......EN.BUILD.STAFF.ARRAYS

This routine cycles through the techniques for each task and feature combination, comparing the total undamaged asset holdings of .STAFF and all of its engineer HQ subordinates with the required equipment for the technique to determine whether or not this command chain can use the technique. This includes all techniques possible for this unit and its subordinates using any combination of equipment available to the group as a whole. The resulting technique array owned by the staff unit indicates the first technique for that task and feature combination the staff can do, or zero if it can use no technique. In this sequence, if damage has been sustained, the staff unit must reassess its capabilities.

Engineer Planning

Engineer planning allows units to request breaching assistance in advance of their actual encounter with known obstacles.

GM.LOCATE.OBSTACLES.IN.UNIT.PATH

This is the lead routine in the code-generated obstacle path point process. Each unit's path is scanned for encounters with obstacles and a path point is created for each.

...EN.PLAN.FOR.OBSTACLES.IN.UNIT.PATH

This routine finds all of the known obstacles located on the unit's path from its present point forward in its direction of movement. For each unbreached obstacle, it calls EN.PLAN.FOR.THIS.OBSTACLE to determine if a planning point is needed.

......EN.PLAN.FOR.THIS.OBSTACLE

This routine determines if a unit needs to plan for the obstacle to be encountered at the given path point. If so, it calls EN.FILE.A.PLANNING.PATH.POINT to determine the planning point.

.........TB.LOCATE.FS.INTERSECTION

This utility routine determines the point at which a path segment crosses a line feature. It is needed because unit encounters with line features keep track of where a unit's center is when the leading edge strikes the obstacle. The location of the leading edge must be calculated.

.........EN.FIND.BREACH.FOR.LINE.OBSTACLE

This routine finds a breach point for the given line obstacle.

.........EN.FILE.A.PLANNING.PATH.POINT

This routine searches back for an amount of time equal to the engineer planning time to find the location of the point at which to plan an engineer mission for the obstacle, creates a ..PLANNING.PT and files it in the unit's path, and creates a mission for the obstacle.

............EN.COMPUTE.TRAVEL.TIME.BETWEEN.TWO.POINTS

This routine is used to determine the location for an engineer planning point. The calling routine steps backward one path segment at a time from the obstacle path point to a dummy path point at the unit's current location.

...........EN.CREATE.A.PLANNING.PATH.POINT

This routine creates a planning path point with the given attribute values.

If a new obstacle is introduced either interactively or by decision tables, a scan of each unit's path determines whether the unit will encounter it. In another case, a previously unknown obstacle may be discovered. In both of these instances, engineers must plan for the encounters.

GM.LOCATE.UNIT.INTERACTION.WITH.OBSTACLE

This routine locates the obstacle path point for every unit that will encounter the given new obstacle.

MF.UNIT.HITS.MINEFIELD

This routine takes care of a unit's encounter with a minefield. If the minefield was previously unknown to the unit's side, all units on the side must examine their paths and plan for encountering the obstacle.

MF.ENTER.OBSTACLE.COMPLEX

This routine takes care of a unit's entrance into an obstacle complex. If the complex was previously unknown to the unit's side, all units on the side must examine their paths and plan for encountering the obstacle.

...EN.PLAN.FOR.OBSTACLE.INTERACTIONS

This routine searches the paths of all maneuver units on the given side for an encounter with the given obstacle, and, on finding such an obstacle path point, creates a planning path point which will generate an engineer mission to breach the obstacle. This routine is called when a new obstacle is introduced either interactively or through decision tables.

......EN.PLAN.FOR.THIS.OBSTACLE

This routine determines if a unit needs to plan for the obstacle to be encountered at the given path point. If so, it calls EN.FILE.A.PLANNING.PATH.POINT to determine the planning point.

......SS.CANCEL.EVENT.FOR.UNIT.IF.SCHEDULED

This routine cancels the event scheduled for a unit. VIC assumes that only one event is scheduled for a unit at any given time, and two unit attributes point to the event and the event type. Two cases are exceptions, the end of a line feature crossing and the end of a minefield delay. The unit event attributes are not set in either of these cases, and such events are not canceled by this routine but must be handled as special cases.

......Schedules a GG.PATH.POINT.ARRIVAL

This is the key event in the ground unit movement process. Units move automatically from path point to path point. A path point arrival initiates a close look at the unit and what it should be doing. In the present case, a moving unit may have added a new path point that is located between its present location and the location of its next event. If so, that event must be canceled and a new event scheduled for the unit.

GG.PATH.POINT.ARRIVAL

...EN.CREATE.MISSION.FOR.A.PLANNING.POINT

This routine creates an engineer mission for the given unit at the given planning point.

......EN.FIND.BREACH.FOR.LINE.OBSTACLE

This routine finds a breach point for the given line obstacle.

......EN.CREATE.A.MISSION.REQUEST

This routine creates an engineer mission for the given data. The assumption is that this is a dynamically generated new mission involving only one feature and that it should be processed immediately. A check is made for duplicate work first, and no new mission is created if the same work is already in the system.

Bridge Demolition

Bridge demolition is controlled by the TB data module and by this external event.

EN.BRIDGE.DEMOLITION

This event processes bridge preparation and demolition requested in an external event file.

...UT.CONVERT.UTM.TO.XY

This utility routine converts UTM coordinates from input data into the XY coordinates used by VIC's processes.

...EN.RETRIEVE.BRIDGING.ASSETS

This routine takes care of finding all retrievable assets owned by engineers at the site of the given bridge feature and returning them to their rightful owners.

...EN.CREATE.A.MISSION.REQUEST

This routine creates an engineer mission for the given data. The assumption is that this is a dynamically generated new mission involving only one feature and that it should be processed immediately. A check is made for duplicate work first, and no new mission is created if the same work is already in the system.

Miscellaneous Processes

Normally, the FL.WG.STRENGTH of a unit indicates its current weapon holdings. For engineer HQ units, however, the FL.WG.STRENGTH only indicates the strength of its assets at its base, not its assets working in the field with work teams. Any RD or LO routine which is trying to determine a supply level or a demand for repairs must be given the HQ's true strength to properly assess requirements.

LO.FIND.STRENGTH
LO.IS.ON.BOARD.FUEL.AND.AMMO.SUFFICIENT
LO.SET.AUTHORIZED.LIMIT
LO.UPDATE.UNITS.SUBSISTENCE.SUPPLIES
RD.ALLOCATE.WPN.REPAIRS
RD.ASSESS.WPN.DEMAND
RD.SET.DEMAND.FLAG

...EN.COMPUTE.ENGINEERS.TRUE.STRENGTH

This routine determines the weapon group strength of an engineer unit in terms of what the unit owns versus what the unit presently has on hand, which is the value of FL.WG.STRENGTH. This routine is used for determining supply and maintenance needs.

......EN.FIND.ENGINEER.HQ

This routine links a GG.UNIT of type ..ENGINEER.UNIT with its associated HQ unit, returning 0 if the unit is not in the set of engineer HQ units.

Defensive positions are now represented as terrain features and no longer correspond to a path point preparation level. Units are protected by defensive positions when they occupy path points and can locate an unoccupied position within their deployment.

GG.PATH.POINT.ARRIVAL

This is the key event in the ground unit movement process. Units move automatically from path point to path point. A path point arrival initiates a close look at the unit and what it should be doing. Several of the engineer sequences are entered through this event.

...EN.LOCATE.DEFENSIVE.POSITION.FOR.UNIT

This routine finds a defensive position of the correct type for a unit at its given path point. If the path point was declared fully prepared by the input data or engineer preparation of the point was requested, then a defensive position already exists for this unit and is pointed to by the GG.PP.POINTER of the path point. Otherwise, a search is made for an unoccupied position of the right type for this unit.

......EN.UPDATE.DP.REPORT

This routine prints a standard format report to the engineer history file so that a defensive position can be traced.

The preparation of defensive positions can be done implicitly and explicitly from the EN module and without any engineer representation using the GG data. Position preparation is the only engineer activity in which the incremental effect must be updated periodically to ensure that the level of protection is correct if a unit comes under attack. This is done during the cyclic update of unit positions.

GG.PERIODIC.POSITION.UPDATE

This event occurs periodically to recalculate unit positions and to update the FEBA, main battle area (MBA), and covering force area (CFA) points.

...EN.UPDATE.ALL.DEFENSIVE.PREPARATIONS.IN.PROGRESS

This routine periodically updates GG.DEF.POSITION.STATUS, GG.UN.PORTION.OF.UNIT.EXPOSED and EN.DP.COMPLETION.FACTOR when a unit is at a path point where defensive preparation is taking place. The update occurs for all cases: explicit engineer preparation, implicit engineer preparation, and unit self-preparation using the unit prototype data.

......EN.CREATE.A.DEFENSIVE.POSITION

This routine creates a defensive position for the given unit at the given path point, filing it in the set of defensive positions.

......TB.FILE.DEFENSIVE.POSITION.IN.GRID

This routine files a newly created defensive position in the set of terrain features of the grid where it is located.

......EN.UPDATE.DP.REPORT

This routine prints a standard format report to the engineer history file so that a defensive position can be traced.

......TB.FIND.GRID.SQUARE.FOR.POINT

This utility routine returns a pointer to the grid cell containing the given location.

......EN.FIND.ENGINEER.HQ

This routine links a GG.UNIT of type ..ENGINEER.UNIT with its associated HQ unit, returning 0 if the unit is not in the set of engineer HQ units.

Engineer missions to prepare positions are generated when the GG data module is read. Each path point may have a specified level of preparation and may be marked for further work by engineers. The positions themselves are now created as terrain features instead of being recorded as path point preparation levels.

GG.FILL.THE.PATH

This routine is part of the loading sequence for the GG data, taking care of reading each of the unit's paths.

...EN.REQUEST.ENGINEER.PREPARATION

This routine creates an engineer mission to prepare the given defensive position.

...EN.CREATE.A.DEFENSIVE.POSITION

This routine creates a defensive position for the given unit at the given path point, filing it in the set of defensive positions.

Decision tables may produce a request for engineers to construct a minefield.

DT.PERFORM.ACTION

...EN.REQUEST.ENGINEER.SUPPORT

This routine issues a request for engineer support of the given unit by requesting a mission of the given task type.

......EN.CONSTRUCT.A.MINEFIELD.MISSION

This routine constructs a minefield mission in support of a unit, which is considered the requesting unit for this mission. This allows minefield missions to be generated by decision tables.

.........MF.CREATE.A.MINEFIELD

This utility routine actually creates a new minefield entity.

.........LO.UPDATE.UNITS.AMMO.SUPPLIES

This routine does the bookkeeping to record the use of ammunition supplies, in this case mines.

......EN.CONSTRUCT.AN.OBSTACLE.MISSION

This routine is a holdover from VIC 2.0, in which it was envisioned that new obstacle mission would be generated by decision table action. As the current EN module was developed, this approach became less feasible and was not implemented. This stub exists because of the possibility of a decision table call.

Implicit engineer work begins as soon as the performing maneuver unit reaches a path point that is within its radius of control of the work site. The work continues as long as the unit keeps the work site within its radius of control. If the path point that the unit is moving toward is not within that distance of the work site, the implicit work is canceled and credit for partial completion of the task is given.

GG.END.OF.WAIT

This event is a key to determining a unit's next action. Like GG.PATH.POINT.ARRIVAL, this event looks at the unit's current situation and schedules its next event.

...EN.STOP.ALL.IMPLICIT.ENGINEER.WORK

This routine causes implicit engineer task performance to end if the performing unit is now moving to a point too far from the work site for the job to continue. Partial completion of each task is computed from the technique being used and the start time attribute of the event.

One of VIC's interactive menu options allows the user to shift weapons from one unit to another. Because of the dual representation of engineer assets, special transfers are required if engineer units are involved in the shifting of weapons.

FL.SHIFT.EQUIPMENT.FROM.UNIT.TO.UNIT

This routine allows the user to shift weapons from one unit to another.

...EN.TRANSFER.ASSETS.HQ.TO.HQ

This routine does all of the bookkeeping involved in transferring engineer assets from one HQ unit to another.

......EN.FIND.ENGINEER.HQ

This routine links a GG.UNIT of type ..ENGINEER.UNIT with its associated HQ unit, returning 0 if the unit is not in the set of engineer HQ units.

......EN.REFILE.ASSET.IN.INVENTORY

This routine takes care of removing an asset from its inventory list and refiling it at the bottom of the section of assets of its type.

......FL.MOUNT.OR.DISMOUNT.A.UNIT

This routine is called when a unit's mounted or dismounted status has changed.

......PT.WRITE.TYPE.10.RECORD

......FL.PRINT.A.UNITS.WPN.GROUP.ATTRIBUTES

......EN.FIND.STAFF.FOR.THIS.HQ

This routine cycles up the command chain from HQ until locating its staff unit.

......EN.BUILD.STAFF.ARRAYS

This routine cycles through the techniques for each task and feature combination, comparing the total undamaged asset holdings of .STAFF and all of its engineer HQ subordinates with the required equipment for the technique to determine whether or not this command chain can use the technique. This includes all techniques possible for this unit and its subordinates using any combination of equipment available to the group as a whole. The resulting technique array owned by the staff unit indicates the first technique for that task and feature combination the staff can do, or 0 if it can use no technique.

VIC provides an interactive menu of user options to monitor progress of the simulation and to allow the user to change certain situations.

SS.GAMER.CONTROL

This routine is the master controller of the interactive menu process.

...EN.PRINT.CONTROL

This routine provides gamer prints of engineer related data.

......EN.PRINT.TASK.MENU

This routine displays the list of task names and the numbers corresponding to them.

.........EN.GET.TASK.NAME

This utility routine associates a text name with each task type being modeled.

......EN.MISSION.DUMP

This routine produces a mission status report for one of the interactive menu options of EN.PRINT.CONTROL if the device is SS.THE.SCREEN and an output mission report if the device is SS.ENGINEER.HISTORY.FILE.

......UT.SELECT.COLOR

......GR.DRAW.CIRCLE

......GR.DRAW.ORIENTED.RECTANGLE
......GR.MOVE
......GR.DRAW
......GR.MAP
......UT.DRAW.CIRCLE
......GR.DRAW.RECTANGLE
......GR.WRITE.TEXT
......UT.FLUSH.GRAPHICS.OUTPUT

......SS.READ.UNIT.NUMBER
This utility routine reads the next input field and returns a unit index.

......EN.FIND.ENGINEER.HQ
This routine links a GG.UNIT of type ..ENGINEER.UNIT with its associated HQ unit, returning 0 if the unit is not in the set of engineer HQ units.

......EN.HQ.DUMP
This routine produces a headquarters status report for one of the interactive menu options of EN.PRINT.CONTROL.

.........EN.GET.TASK.NAME
This utility routine associates a text name with each task type being modeled.

......EN.WT.DUMP
This routine produces a work team status report for one of the interactive menu options of EN.PRINT.CONTROL.

.........GM.PRINT.CMBT.STATUS
This utility routine associates a character string with each combat status index and prints that string to the current output file.

.........EN.GET.TASK.NAME
This utility routine associates a text name with each task type being modeled.

......EN.JOB.DUMP
This routine produces a job status report for one of the interactive menu options of EN.PRINT.CONTROL.

.........EN.GET.TASK.NAME
This utility routine associates a text name with each task type being modeled.

The engineer postprocessor requires certain key information about the scenario to build its data structures and to process the engineer history file. This information is printed either in the engineer history file or the engineer setup file at the end of each run.

SS.STOP.SIMULATION
This routine is called to take appropriate actions to stop the simulation.

...EN.PRINT.FINAL.REPORT

This routine takes care of printing all necessary output to the engineer history file and setup file at the end of the simulation.

......EN.MISSION.DUMP

This routine produces a mission status report for one of the interactive menu options of EN.PRINT.CONTROL if the device is SS.THE.SCREEN and an output mission report if the device is SS.ENGINEER.HISTORY.FILE.

LIST OF ACRONYMS

AMIP	Army Model Improvement Program
ATD	antitank ditch
AVLB	armored vehicle launched bridge
BAI	battlefield air interdiction
CASTFOREM	Combined Arms and Support Task Force Evaluation Model
CFA	covering force area
CSS	combat service support
EFAM	engineer functional area model
EMIP	Engineer Model Improvement Program
EN	Engineer
ENG	type engineer
FACE	forward aviation combat engineering
FARPS	forward arming and refueling points
FEBA	forward edge of battle area
FL	Front-Line Attrition
FORCEM	Force Evaluation Model
GG	Global Ground
GM	Ground Movement
HQ	headquarters
LO	Logistics
MBA	main battle area
MF	Minefield
MSR	main supply route
PLT	level platoon
SS	System Specification
TB	Terrain-Barrier
TRADOC	U.S. Army Training and Doctrine Command
USACERL	U.S. Army Construction Engineering Research Laboratory
USAEWES	U.S. Army Engineer Waterways Experiment Station
VIC	Vector-In-Commander

DISTRIBUTION

Chief of Engineers
ATTN: CEHEC-IM-LH (2)
ATTN: CEHEC-IM-LP (2)
ATTN: CERD-L
ATTN: CERD-M
ATTN: DAEN-ZA

US Army School 65473-5000
ATTN: ATSE-CDC-M

TRADOC Analysis Command
ATTN: ATRC-FM
ATTN: ATRC-WEA

US Army Model Improvement and Study Management Agency (MISMA)
ATTN: ODUSA-OR

Fort Belvoir, VA 22060
ATTN: CERSC-MD
ATTN: Engr Studies Center Library
ATTN: Engr Library
ATTN: CECC-R

CEWES Vicksburg, MS 39180
ATTN: Library
ATTN: CEWES-GV-A
ATTN: CEWES-EN-A
ATTN: CEWES-GM-L

USAMSAA
Aberdeen Proving Grounds, MD 21005-5071
ATTN: AMXSY-CC

CECRL 03755
ATTN: Library

NCEL 93043
ATTN: Library

US Military Academy 10996
ATTN: Dept of Military Science

US Army Command & General Staff College 66027
ATTN: ATZL-SWH

US Army Armor School 40121
ATTN: ATSB-DOTD-CBT

US Government Printing Office 20401
Receiving/Depository Section (2)

US Army Combined Arms Center 66027-5300
ATTN: ATZL-CAS

Army War College 17013
ATTN: Library

Defense Logistics Studies Information Exchange
ATTN: AMXMC-D

Defense Technical Info. Center 22304
ATTN: DTIC-FAB (2)